University of Sussex

# Motion Control of Unmanned Ground Vehicle Using Artificial Intelligence

A Thesis Submitted for the Degree of Doctor of Philosophy

By

**Auday Basheer Essa Al-Mayyahi**

B.Sc. (Hons), M.Sc. (Hons)

Department of Engineering and Design

School of Engineering and Informatics

University of Sussex

Brighton

United Kingdom

**May 2018**

# Dedication

*To my parents, brothers and sisters. To my beloved wife, Miaad and my adorable children.*

# Abstract

The aim of this thesis is to solve two problems: the. trajectory tracking and navigation, for controlling the motion of unmanned ground vehicles (UGV). Such vehicles are usually used in industry for assisting automated production process or delivery services to improve and enhance the quality and efficiency.

With regard to the trajectory tracking problem, the main task is to design a new method that is capable of minimising trajectory-tracking errors in UGV. To achieve this, a comprehensive mathematical model needs to be established that contains kinematic and dynamic characteristics beside actuators. In addition, different trajectories need to be generated and applied individually as a reference input, i.e. continuous gradient trajectories such as linear, circular and lemniscuses or a non-continuous gradient trajectory such as a square trajectory. The design method is based on a novel fractional order proportional integral derivative (FOPID) control strategy, which is proposed to control the movement of UGV to track given trajectories. Two FOPID controllers are required in this design. The first FOPID is constructed in order to control the orientation of UGV. The second FOPID controller is to control the speed of UGV. The particle swarm optimization (PSO) algorithm is used to obtain the optimal parameters for both controllers. The significance of the proposed method is that an observable improvement has been achieved in terms of minimising trajectory-tracking errors and reducing control efforts, especially in continuous gradient trajectories. The stability of the proposed controllers is investigated based upon Nyquist stability criterion. Moreover, the robustness of the controllers is examined in the presence of disturbances to demonstrate the effectiveness of the controllers under certain harsh conditions. The influence from external disturbances has been represented by square pulses and sinusoidal waves. The drawback of this method, however, a highly trajectory tracking error is observed in non-continuous gradient trajectories due to the sharpness of the rotation at the corners of a square trajectory.

To overcome this drawback, a new controller, abbreviated as (NN-FOPID), has been proposed based on a combination of neural networks and the FOPID. The purpose is to minimise the trajectory tracking error of non-continuous trajectories, in particular. The Levenberg-Marquardt (LM) algorithm is used to train the NN-FOPID controller. The neural networks' cognitive capacities have made the system adaptable to respond effectively to the variants in trajectories. The obtained results by using NN-FOPID have shown a significant improvement of reducing errors of trajectory tracking and increasing control efforts over the results by FOPID.

The other task is to solve the navigation problem of UGV in static and dynamic environments. This can be conducted by firstly constructing workspace environments that contain multiple dynamic and static obstacles. The dynamic obstructing obstacles can move in different velocities. The static obstacles can be randomly positioned in the workspace and all obstacles are allowed to have different sizes and shapes. Secondly, a UGV can be placed in any initial posture on the condition that it has to reach a given destination within the boundaries of the workspace. Thirdly, a method based on fuzzy inference systems (FIS) is proposed to control the motion of the UGV. The design of FIS is based on fuzzification, inference engine and defuzzification processes. The navigation task is divided into obstacle avoidance and target reaching tasks. Consequently, two individual FIS controllers are required to drive the actuators of the UGV, one is to avoid obstacles and the other is to reach a target. Both FIS controllers are combined through a switching mechanism to select the obstacle avoidance FIS controller if there is an obstacle, otherwise choosing reaching target FIS. The simulation results have confirmed the effectiveness of the proposed design in terms of obtaining optimal paths with shortest elapsed time.

Similarly, a new method is proposed based on an adaptive neurofuzzy inference system (ANFIS) to guide the UGV in unstructured environments. This method combines the advantages of adaptive leaning and inference fuzzy system. The simulation results have demonstrated adequate achievements in terms of obtaining shortest and feasible paths whilst avoiding static obstructing obstacles and hence reaching the specified targets speedily.

Finally, a UGV is constructed to investigate the overall performance of the proposed FIS controllers practically. The architecture of the UGV consists of three ultrasonic sensors, a magnetic compass and two quadratic decoders that they are interfaced with an Arduino microcontroller to read the sensory information. The Arduino, who acts as a slave microcontroller is serially connected with a master Raspberry Pi microcontroller. Raspberry Pi and Arduino communicate with each other based on a proposed hierarchical algorithm. Three case studies are introduced to demonstrate the effectiveness and the validation of the proposed FIS controllers and the UGV’s platform in real-time.

# Acknowledgment

I would first to thank my supervisors, Dr. William Wang and Dr. Phil Birch for their supervision, support, open door policy. Without them, the advancement of this work could not have been achieved. I am very grateful for their supervision and guidance.

My thanks are extended to the Ministry of Higher Education and Scientific Research in Iraq for the grant as well as the University of Basra. My deepest gratitude goes to Dr. Habeeb Jaber Nekad - Dept. of Electrical Engineering for his long-term support while I was away. My special thanks go to Dr. Ramzy Salim Ali - Head of Electrical Engineering Dept. for his support and advice.

I would like to thank my colleagues Sola Ajiboye, Sultan Almazroui, Alaa Husssien, Iniabasi Ituen, Eze Chinedu, Chris Johnson, Abdurrhman Alroqi, Junlong Duan and Tabassum Qureshi for their friendship.

Another people I met outside my academic circle who have befriended and encouraged me during my stay in UK. In particular, I would to thank mention Dr. Abbas Ali and Dr. Bassam Ashoor for their steady help and support since my arrival to the UK until the last moment.

In addition, I would to thank my examiners, Prof. Hongnian Yu and Prof. Chris Chatwin, for reviewing my work and for their constructive feedback.

Last but certainly not least, I would express a deep sense of gratitude to my parents, siblings, my wife and my children for their continuous love and support, without them this work would not have been possible.

# CONTENTS

# List of Abbreviations

| Item No. | Abbreviation | Description |
|---|---|---|
| 1 | UGV | Unmanned Ground Vehicle(s) |
| 2 | UAV | Unmanned Aerial Vehicle(s) |
| 3 | USV | Unmanned Surface Vehicle(s) |
| 4 | UUV | Unmanned Underground Vehicle(s) |
| 5 | GPP | Global Path Planning |
| 6 | LPP | Local Path Planning |
| 7 | DoF | Degree(s) of Freedom |
| 8 | PID | Proportional Integral Derivative |
| 9 | PD | Proportional Derivative |
| 10 | PI | Proportional Integral |
| 11 | NN | Neural Network(s) |
| 12 | ANN | Artificial Neural Network(s) |
| 13 | GPP | Global Path Planning |
| 14 | LPP | Local Path Planning |
| 15 | APF | Artificial Potential Field |
| 16 | FLC | Fuzzy logic controller |
| 17 | EA | Evolutionary Algorithm |
| 18 | GA | Genetic Algorithm |
| 19 | SA | Simulated Annealing |
| 20 | PSO | Particle Swarm Optimization |
| 21 | ACO | Ant Colony Optimization |
| 22 | HACO | Heterogeneous Ant Colony Optimization |
| 23 | SLAM | Simultaneous Localisation and Mapping |
| 24 | ICC | Instantaneous Centre of Curvature |
| 25 | 2D | Two dimensions |
| 26 | DC | Direct Current |
| 27 | FOPID | Fractional Order Proportional Integral Derivative |
| 28 | ISE | Integral Square of Error |
| 29 | MIMO | Multi-Input Multi-Output |
| 30 | SISO | Single Input Single Output |
| 31 | BP | Back-Propagation |
| 32 | LM | Levenberg-Marquardt |
| 33 | MSE | Mean Square Error |
| 34 | FIS | Fuzzy Inference System(s) |
| 35 | CoA | Centre of Area |
| 36 | LD | Left Distance |
| 37 | FD | Front Distance |
| 38 | RD | Right Distance |
| 39 | OA | Obstacle Avoidance |
| 40 | TR | Target Reaching |
| 41 | AD | Angle Difference |

| | | |
|---|---|---|
| 42 | OS | Obstacle Sensing |
| 43 | ANFIS | Adaptive Neuro-Fuzzy Inference System |
| 44 | RMSE | Root Mean Square Error |
| 45 | MISO | Multiple Input Single Output |
| 46 | SIMO | Single Input Multiple Output |
| 47 | 3D | Three Dimensions |
| 48 | USB | Universal Serial Bus |
| 49 | I2C | Inter-Integrated Circuit |
| 50 | SCL | Serial Clock |
| 51 | SDA | Serial Data |
| 52 | MSB | Most Significant Bit |
| 53 | PWM | Pulse Width Modulation |
| 54 | GPIO | General Purpose Input/output |
| 55 | SPI | Serial Peripheral Interface |
| 56 | TTY | Teletypewriter |

# List of Symbols

| Symbol | Description | Unit |
|---|---|---|
| $L$ | Distance between right and left wheels | m |
| $v_l$, | Linear velocity of left wheel | m/s |
| $v_r$ | Linear velocity of right wheel | m/s |
| $(x_o, y_o)$ | Starting point coordinates | m |
| $(x_g, y_g)$ | Target point coordinates | m |
| $(x_c, y_c)$ | Actual coordinates of vehicle | m |
| $v_x$ | Longitudinal velocity of vehicle | m/s |
| $v_y$ | Lateral velocity of vehicle | m/s |
| $a_x$ | Longitudinal acceleration of vehicle's centre of mass | m/ $s^2$ |
| $a_y$ | Lateral acceleration of vehicle's centre of mass | $m/s^2$ |
| $F_{xl}$ | Longitudinal force exerted on left wheel | N |
| $F_{xr}$ | Longitudinal force exerted on right wheel | N |
| $F_{yl}$ | Lateral force exerted on right wheel | N |
| $F_{yr}$ | Lateral force exerted on right wheel | N |
| $\theta$ | Orientation of UGV | rad |
| $\omega_r$ | Angular velocity of right wheel | rad/s |
| $\omega_l$ | Angular velocity of left wheel | rad/s |
| $\omega$ | Angular velocity of vehicle | rad/s |
| $S_r$ | Travelled distance of right wheel | m |
| $S_l$ | Travelled distance of left wheel | m |
| $S$ | Travelled distance of vehicle | m |
| $v$ | Velocity of the vehicle | m/s |
| $a$ | Distance between the centre of mass and driving wheels' axis in x-direction | m |
| $r$ | Radius of vehicle's wheel | m |
| $m$ | Mass of vehicle with driving wheels and DC motors | kg |
| $I_c$ | Moment of inertia about the centre of mass | $Kg.m^2$ |
| $q$ | Vector of generalized position coordinates | - |

| | | |
|---|---|---|
| $\dot{q}$ | Vector of longitudinal and angular velocities of generalized coordinates. | - |
| $M(q)$ | Symmetric positive definite inertia matrix of the system, | - |
| $C$ | Centripetal and Coriolis forces matrix, | - |
| $F(\dot{q})$ | Surface friction matrix, | - |
| $G(q)$ | Gravitational vector, | - |
| $B(q)$ | Input transformation matrix, | - |
| $T$ | Input transformation matrix and input vector. | - |
| $\omega_m$ | Angular speed of motor | rad/s |
| $i_a$ | Motor current | A |
| $E_a$ | Applied voltage to motor | V |
| $v_b$ | Back electromotive force (e.m.f) voltage | V |
| $R_a$ | Resistance of armature winding | Ω |
| $L_a$ | Inductance in motor winding | H |
| $J_a$ | Motor inertia | Kg.m$^2$ |
| $k_m$ | Torque constant | N.m/A |
| $k_b$ | Back electromotive force constant | V.s/rad |
| $B$ | Viscous friction | N.m.s |
| $k$ | Iteration number | - |
| $T_m$ | Motor torque | N.m |
| $T_L$ | Load torque | N.m |
| *Iter* | Current number of iterations | - |
| $Iter_{max}$ | Maximum number of iterations | - |
| $D$ | Dimension search space | - |
| $S_i^k$ | Current position of particle i$^{th}$ at iteration k | - |
| $V_i^k$ | Current velocity of particle i$^{th}$ at iteration k | - |
| $pbest_i$ | Local best position visited by i$^{th}$ particle | - |
| $gbest$ | Global best position visited by a warm | - |
| $W$ | Inertia weight function | - |
| $W_{max}, W_{min}$ | Maximum and minimum values of inertia weight | - |

| | | |
|---|---|---|
| $\alpha$ | Learning rate or it is called the step size | - |
| $N$ | Number of weights | - |
| $J$ | Jacobian matrix | - |
| $H$ | Hessian matrix | - |

# List of Figures

# List of Tables

# List of Publications

## Journal Publications

[1] **Al-Mayyahi, Auday**, Weiji Wang, Alaa Hussein, Phil Birch, (2017) *Motion control design for unmanned ground vehicle in dynamic environment using intelligent controller.* International Journal of Intelligent Computing and Cybernetics, Vol. 10 Issue: 4, pp.530-548.

[2**] Al-Mayyahi, Auday**, Wang, William and Birch, Philip (2016) *Design of fractional-order controller for trajectory tracking control of a non-holonomic autonomous ground vehicle.* Journal of Control, Automation and Electrical Systems, 27 (1). pp. 29-42. ISSN 2195-3880.

[3] **Al-Mayyahi, Auday**, Wang, Weiji and Birch, Philip (2015) *Levenberg-Marquardt optimised neural networks for trajectory tracking of autonomous ground vehicles.* International Journal of Mechatronics and Automation, 5 (2/3). pp. 140-153. ISSN 2045-1067.

[4] **Al-Mayyahi, Auday**, Wang, William and Birch, Philip (2014) *Adaptive neuro-fuzzy technique for autonomous ground vehicle navigation.* Robotics, 3 (4). pp. 349-370. ISSN 2218-6581.

## Conference Proceedings

[1] **Al-Mayyahi, Auday**, Wang, William and Birch, Phil (2015) *Path tracking of autonomous ground vehicle based on fractional order PID controller optimized by PSO.* In: 2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI), 22-24 Jan. 2015, Herl'any.

[2**] Al-Mayyahi, Auday** and Wang, William (2014) *Fuzzy inference approach for autonomous ground vehicle navigation in dynamic environment.* In: 2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), 28-30 November 2014, Batu Ferringhi.

# Chapter 1
# Introduction

## 1.1 Background

Unmanned ground vehicles (UGV) can be defined as programmable and multi-purpose machines, capable of gathering and extracting information from its surroundings using sensory devices and control units to plan and execute its mission within its environment without human interventions. The types of unmanned vehicles can be classified according to its working environment into three types: aerial, terrestrial and aquatic vehicles. In an unmanned aerial vehicle (UAV), the operating workspace is simply the air. The aquatic type also is classified into two sub-categories; when a vehicle operates on the surface of water, it is called an unmanned surface vehicle (USV); when the vehicle operates under the water, it is called an unmanned underwater vehicle (UUV). The final type is terrestrial vehicles which they can be any type wheeled unmanned ground vehicles.

Nowadays, there is an obvious increasing of such vehicles in many applications of industrial automation and daily life. The applications of such vehicles can be summarised based on the operating workspace. For instance, in a hostile or military environment, it would be essential under particular circumstances to have a UGV for performing some predefined missions where it is dangerous or inconvenient to achieve such missions by humans. In addition, for mining purposes in some areas which might contain serious radioactivity or poisonous gases. Hence, the application of a UGV will be the key means for such workspaces. The UGV can be utilised in many other environments, such as those for increasing productivity, reducing costs and improving life quality. They have further potential applications in industrial production lines and transportation systems. They can be utilised in a transportation system and performing special functions in manufacturing processes such as loading and unloading of equipment and products. Recently, they have been employed in advanced security applications for surveillance purposes.

Designing a UGV requires a knowledge from many disciplines of engineering, including mechanical, electrical, and computer engineering. A UGV consists mainly of components such as sensors, intelligent controllers and actuation systems. Those components are the key element of achieving the autonomy and make a UGV capable of working and performing its tasks without human assistance. Hence, a UGV operates autonomously based on its perception,

intelligence, and action. The efficient navigation of such vehicles is achieved via continuous interactions among perception, intelligence and action. A UGV is capable of understand and interacting with its surroundings in a compact approach to reach its specific targets smoothly. The perception can be obtained based on utilising sensing devices that will enable a UGV to sense and understand objects in order to avoid collision and operate in a safe manner. The artificial intelligence is the key element of a decision-making. This will make a UGV behaves intelligently in its workspace to perform the required tasks. Many controllers and algorithms can be used to obtain an intelligent technique that makes a UGV learning and inferencing within the workspace environments.

## 1.2 Research Motivation

In industrial and manufacturing automation, the safety, productivity, accuracy are essential factors when the performance of a process is evaluated. The applications of UGV have significantly increased and they become ubiquitous transportation system and even a major tool for the development of industrial automation. The motivation comes from the research work presented in the state of the art literature to solve the trajectory tracking and navigation problems for non-holonomic robotic systems. These problems are particularly vigorous because of their challenges in terms of theoretical nature and practical importance. The theoretical behaviour of non-holonomic constrains presents a number of challenges. For instance, such systems are underactuated, that is, the number of control inputs is less than the number of states or variables of the system to be controlled. Thus, motion planning implies that the systems can be completely controlled with a fewer number of actuators, thereby improving the overall cost-effectiveness of the system. In addition, underactuation can provide backup control techniques for a fully actuated system.

Recently, trajectory tracking and navigation problems of unmanned ground vehicles have become popular and vigorous research topics. Additionally, such problems are within the main challenges in the development of advanced UGV. The primary aim of this research is to utilize intelligent controllers and optimization algorithms to be used in solving the trajectory tracking and navigation problems. In this thesis, trajectory tracking and navigation are intensively studied based on different scenarios to develop a fully autonomous navigation. These two separate problems need to be addressed to improve the industrial environments. The firstly problem is the trajectory tracking. In this problem, a UGV is required to follow a specific trajectories such as continuous gradient trajectories and non-continuous gradient trajectories.

The second problem is the navigation. The aim is to make a UGV capable of traversing between an initial position and a target position without colliding with obstructing obstacles including dynamic and static obstacles and reach its destination safely and efficiently.

## 1.3 Research Objectives

The fundamental task of an unmanned ground vehicle is to design intelligent robust controllers, which enable the UGV to track pre-defined trajectories and to navigate efficiently in a complex environment populated with static and dynamic obstacles. In the trajectory tracking, the UGV has to track given trajectories accurately and minimise trajectory-tracking error. In addition, the UGV must create a collision free path and avoid obstacles that might obstruct its path. The UGV must be capable of searching another feasible path when encountering obstacles that blocking its way, optimal and shortest paths should be obtained always in reasonable time.

The author's objectives of research work can be divided into categories based on the reported research topics as follows:

**a) Trajectory tracking**

The main aims are to minimise trajectory-tracking error and reduce the control efforts in any of utilised continuous and non-continuous gradient trajectories. To achieve the aims, several objectives need to be conducted as summarised below:

1) To generate continuous gradient trajectories such as lemniscate, circular, and linear trajectories and non-continuous gradient trajectories such as square trajectories.
2) To model the UGV based on non-holonomic kinematic characteristic, dynamic and actuators units.
3) To introduce intelligent controllers based on the fractional order PID controllers and neural networks to demonstrate better performance in terms of minimising trajectory tracking error and improve control efforts. Two controllers are need to govern the orientation and the speed of a UGV.
4) To optimise the parameters of the designed controllers to enhance the overall responses using the particle swarm optimisation and Levenberg–Marquardt algorithms using online training in the feedback control systems.
5) To build and simulate the proposed controllers and the model of the UGV to examine the responses comparing to the state of the art methodologies.

**b) UGV Navigation**

This problem comprises the obstacle avoidance and target reaching. Thus, the aim is to keep the UGV away from any obstructing bodies and find a collision-free path from an initial position to a destination position where the navigation will be stopped when a required target is reached. To achieve that, the following tasks are needed:

1) To build a navigation platform that contains obstructing static and dynamic obstacles, which move randomly with different speeds and orientations.
2) To provide sensing technology to sense the surrounding and localise the UGV within its workspace to find a collision-free path between starting and targeting positions.
3) To design intelligent controllers to drive the motion of the UGV efficiently in unknown environments. The controllers are combined by a switching mechanism, to achieve the obstacle avoidance and target reaching tasks as needed in working-scenarios.
4) To combine and simulate the navigation platform, the model of UGV, sensing schemes and the proposed controllers i.e. a fuzzy inference system and an adaptive neuro-fuzzy inference system.
5) To practically implement the architecture of the UGV to validate the overall performance of the proposed intelligent controllers.

## 1.4 Contributions to Knowledge

The work of this thesis can be understood by considering two main research questions. Firstly, the problem of trajectory tracking of an unmanned ground vehicle is studied based on different predefined trajectories. These trajectories have been generated based on two categories i.e. the continuous and non-continuous gradient trajectories. Examples of continuous gradient are linear, circular and lemniscate trajectories whereas an example of non-continuous gradient is a square trajectory. The novelty and contribution of this research lie in the following tasks:

1) To extend and develop a model of UGV to take into consideration the non-holonomic constraints.
2) To introduce novel fractional order proportional integral derivative (FOPID) controllers to tackle the problem of trajectory tracking. The parameters of the FOPID controllers are optimally tuned using a particle swarm optimisation (PSO) algorithm. The obtained results have demonstrated an observable development for minimising the trajectory tracking error and reducing control efforts. Thus, this in turn leads to improve the

operational response of the system in continuous and non-continuous gradient trajectories. However, the improvement has not been satisfied with the non-continuous gradient trajectory at this stage.

3) Stability analysis of the proposed FOPID controllers and the controlled plant by obtaining the Multi-Input Multi-Output (MIMO) and Single-Input Single-Output (SISO) systems, then apply Nyquist stability criterion, which is based on Cauchy's theorem to determine the stability of the system.
4) Robustness investigation in the presence of disturbances to verify the effectiveness of the proposed methodology into different operating conditions.
5) To propose new controllers based on the FOPID controllers and neural networks (NN) to create a new architecture called NN-FOPID model, which is designed to combine the advanced learning capabilities of neural networks with the FOPID controllers. The obtained results using NN-FOPID model have been demonstrated significant improvements for minimizing the tracking error between desired and actual trajectories of the non-continuous gradient trajectory.

Secondly, the problem of navigation of the UGV is investigated into static and dynamic environments. The contribution of this research lies in the following tasks:

1) To create a workspace topology that contains randomly moving objects with different speeds and orientations. The moving objects are dimensioned with different shapes and sizes.
2) To present fuzzy inference systems to control the motion of the UGV in dynamic environments without colliding with obstructing obstacles and then to reach a specified destination.
3) To propose another control methodology based on an adaptive neuro fuzzy inference system to investigate the performance of different control systems in newly constructed environments.
4) Finally, a UGV architecture has been implemented based on the author's design. Real time experiments have been conducted to investigate the overall performance of the proposed methodology i.e. FIS controllers. In the design, new modules of embedded system parts are interfaced with each other. Thus, algorithms for communication protocols are introduced in order to enable the devices communication with each other. The experimental results have been successfully and feasibly shown that the proposed methodology is efficiently performed the obstacle avoidance and target reaching.

## 1.5 Research Application

In industrial automation; the safety, productivity, accuracy are essential factors when the efficiency of a process is evaluated. Hence, applications of UGV can be significantly beneficial for development of a workspace automation. The UGV can be used in industry for moving equipment and delivery services from production lines to storage areas.

The automation includes several advantages rather than manual approaches in terms of solving some problems and challenges as follows:

1) A lot of time and an enormous amount of energy are required to transport equipment. Therefore, time, resources and energy can be saved and optimally managed.
2) A huge cost is required to employ enough store assistants that could be a challenging for growing industries and firms. Profit will also be maximised by reduction a cost that would be spent on frequent staff training and continuous wage payments.
3) The repetitive nature of delivering services can be tedious for workers, thus, automation would be a solution for employers in terms of labouring.
4) Health and safety are crucial to involved individuals due to moving heavy equipment. Hence, associated risks will be reduced.
5) Reaching some positions might be difficult or hazardous, therefore, the automation can an efficient solution for increasing quality and efficiency of industry.

According to all above challenges, the employing of UGV in the industrial environments is desirable.

## 1.6 Structure of Thesis

Efforts have been made in this thesis to develop robust trajectory tracking and navigation methodologies for unmanned ground vehicles that can be deployed in different industrial automation scenarios to facilitate automatic parts and equipment transportation in a line of manufacturing and production process. Different trajectory tracking algorithms have been studied in this regard and techniques have been proposed to tackle the associated problems.

The thesis is divided into two parts. The first part deals with trajectory tracking problem based on different patterns of trajectories. The second part investigates the navigation and obstacle avoidance problem.

Initially, Chapter 2 explains a background of some key elements in this thesis and reviews intensively previous work in the literature.

The actual research work begins in Chapter 3 by analysing of the mathematical model of a spatial unmanned ground vehicle subject to non-holonomic constraints. The energy-based Lagrangian approach is deployed in modelling of the UGV. In addition, it describes the basic properties of non-holonomic and holonomic systems. Chapter 4 is devoted to the proposed novel approach to solve a trajectory-tracking problem based on the fractional order PID controllers. This controller has five parameters. The best values of those parameters are determined based on the particle swarm optimization.

Chapter 5 introduces the design of novel fractional order PID neural network controllers to compare the obtained trajectory tracking error by using fractional order PID controllers. Chapter 6 is dedicated for implementation of the developed fuzzy inference systems (FIS) for achieving obstacle avoidance and destination reaching. In this chapter, the core of building the model is presented to explain the designing process of two fuzzy inference systems for a dynamic environment that contains randomly moving obstacles. Chapter 7 presents a new controller for solving the navigation problem based on adaptive neuro fuzzy inference systems. Different scenarios are investigated to take into a consideration the adaptive performance of another approach for different operating conditions.

In Chapter 8, vehicle experiments have been introduced; it demonstrates the architecture of an unmanned ground vehicle. The embedded sensors are illustrated to describe how the UGV perceives the surroundings in a dedicated workplace. Additionally, the proposed FIS given in Chapter 6 is implemented in this chapter practically to validate the overall performance. Finally, Chapter 9 introduces with a brief summary the conclusions of the research work. The work has answered some research questions and but has raised new questions. Accordingly, the thoughts and suggestions for a number of issues that can be future directions of research are stated.

# Chapter 2

# Literature Review

## 2.1 Introduction

This chapter provides information about the relevant work related to the research topics. The concentration is based on exploring a variety of techniques that used in trajectory tracking and navigation solutions. A target reaching and obstacle avoidance are the essential components for performing the navigation of UGV. This makes the UGV interacts with its surrounding and perceives its workspace environment. In the last three decades, there was a considerable amount of research conducted on the development of an efficient navigation system and motion-planning algorithm in structured and unstructured environments with many applications.

The motion planning of the UGV can be classified and analysed into two main research areas i.e. trajectory tracking and navigation. These two areas contain various control strategies and methodologies. Many intelligent controllers have been utilised for solving such the problems such as proportional integral derivative controller, fuzzy logic control, neural networks and hybrid controllers that combine more than one controller. They have been widely applied in dealing with motion planning in a variety of case studies and scenarios based on different environments. In additional, many evolutionary optimization algorithms have been applied to solve the motion planning. The trajectory tracking is considered based on generating different reference trajectories. Hence, the UGV should be capable of tracking the desired trajectories with minimum tracking error.

The main goal of this research is to propose novel controllers based on different methodologies for controlling the trajectory tracking of the UGV. The introduced controller is compared with literature to analyse the advantage and disadvantage of the new methodologies.

The navigation is a task of path planning and obstacle avoidance into static and dynamic environments. This can be achieved based on many approaches and methodologies such as an artificial potential field, visibility graphs, a genetic algorithm, a simulated annealing, a particle swarm optimization and an ant colony optimization technique. It has been noticed that the most of the conducted work has been applied to static environments, which this does not imply sophisticated behaviour of the UGV to avoid obstacles and reach its destination. Hence, there is a need for investigating such techniques and optimization algorithms in cluttered dynamic

environments. Although some techniques of the existing works had shown the applications of UGV in dynamic environments, some limitations and constrained were assumed to ease the navigation process such as obstacles move at constant speed, the size and shape of obstacles are identical. The intelligent controllers such as artificial intelligence in a dynamic environment where obstacles move randomly with various speeds and directions are utilised by the author. Based on real world scenarios, the obstacles are also considered to be constructed based on a variety of sizes and shapes. Such considerations will cover all the potential challenges that the UGV might confront whilst it navigates in such sophisticated workspaces. The intelligent controllers will endow the UGV intellectual abilities to execute its manoeuvrability safely and efficiently.

## 2.2 Chapter Organisation

The chapter is organised as follows: In the following section, a brief overview of a control scheme for UGV is presented. In Section 2.4, the trajectory-tracking problem of is reviewed. In Section 2.5, an intensive literature review is conducted for navigation of UGV based on different techniques and methods. Finally, chapter summary is described in Section 2.6.

## 2.3 Control Scheme of Unmanned Ground Vehicles

Trajectory tracking and navigation in an unknown environment can be achieved by local reactive path planning approach using an on-board sensory information. Fig. 2.1 depicts the general control scheme for the main concepts of the trajectory tracking and navigation architecture. It can be clearly seen that a UGV requires several elements to accomplish the vehicle's navigation. For instance, several sensors are used to sense the surroundings of the UGV. Many control techniques can be utilised to identify the localisation of UGV within its workspace. In trajectory tracking, a desired trajectory must be tracked accurately based on a motion control approach. In addition, the UGV must decide how to act for achieving its goals without collision with surrounding obstacles and finally, the UGV must modulate its wheels' steering to obtain an optimal and a desired path towards a destination point. The navigation will be efficient and successful if the elements of navigation scheme can be verified (Siegward et al., 2011).

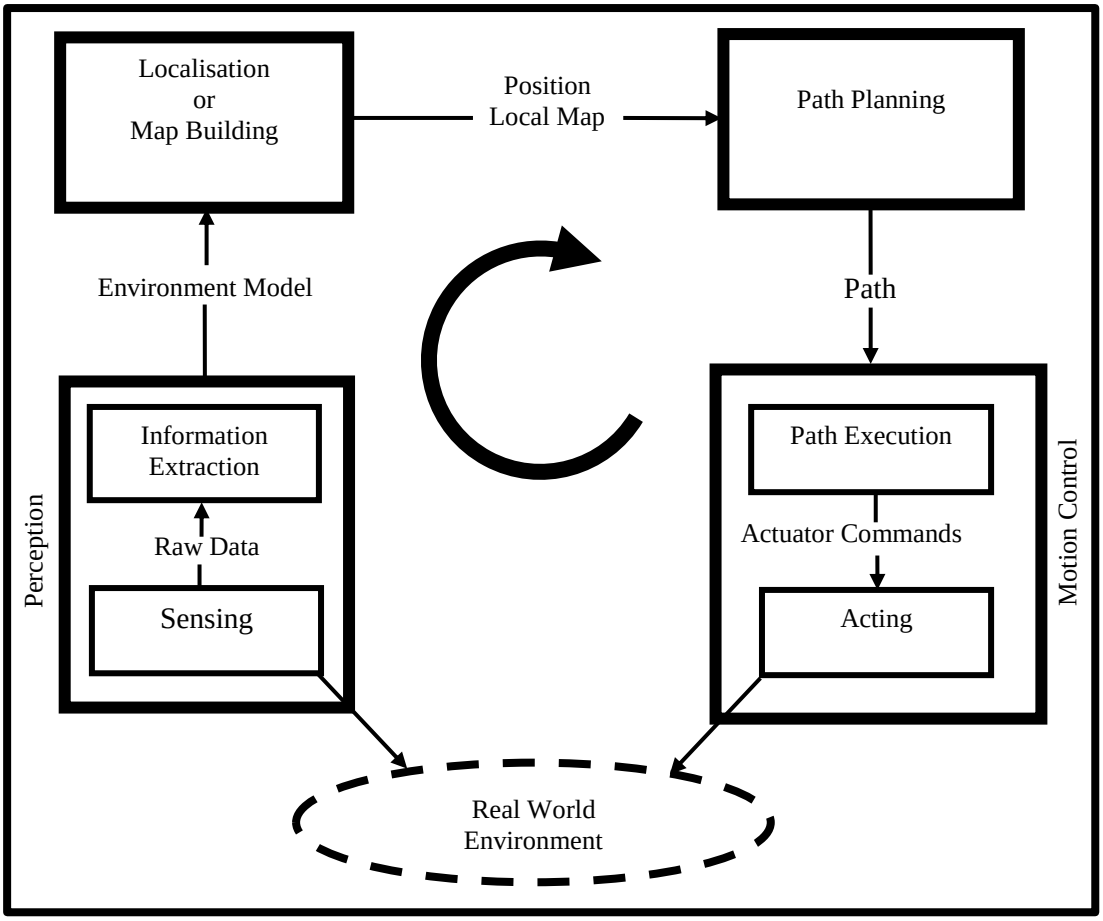



Fig. 2.1 General Control scheme for UGV.

### *2.3.1 Path Planning*

Path planning is an important and primitive step for unmanned ground vehicles to find the optimal path between starting and destination points. The navigation missions involve the activities of stopping, turning and running repeatedly. Hence, optimal paths should be obtained to minimise the amount of turning and braking based on a specific application. Path planning requires a map of the environment and the UGV must be capable of localising its posture with respect to the map. Consequently, several questions have been raised, such as how to create a map for a workspace; how can UGV localise itself into the workspace, and how can the UGV avoid obstacles and deal with uncertain position information.

Path planning is a vital task in the design of UGV. In addition, it is one of the most studied topics. The definition of path planning is to find a feasible route in which kind of environments after avoiding obstacles. The Methods of path planning can be classified into two categories; global path planning (GPP) and local path planning (LPP) (Sedighi et al., 2004) according to the characteristics of the environment. In GPP, the vehicle is simulated based on a static

environment which the vehicle has a prior knowledge about its work environment and the features are completely known and the terrain should be static. Hence, the vehicle can navigate smoothly between an initial position and the desired whilst avoiding obstacles. Whereas, in LPP, the vehicle does not have a prior information and it might work in unknown environment. In other words, LPP is required if the environment is dynamic and features of which are partially or entirely unknown. Therefore, the vehicle must use its own sensors to extract the required information to achieve save navigation and to reach to its target.

Path planning is a fundamental element to make the vehicle is fully autonomous and reliable. Its goal is to plan a sequence of suitable paths via multiple points and segments subjected to some techniques and optimization criteria that allows the vehicle to complete its task objectives by reaching the specified destination point from the starting location with free collision route. Path planning involves several methods such as Visibility Graph (Berg et al., 2000), Voronoi Diagram (Garrido et al., 2011), Cell Decomposition and Bug Algorithm (Choset et al., 2005).

The map representation is the key step of achieving the path planning. In order to plan a path, a workspace environment is represented either discretely or continuously. The discrete representation is the most dominated based on a computer because it divides the map into an equilateral grid or a hexagonal structure to create a topological map. The spatial coordinates of each vertex of the grid are known and they can be stored computational into a repository of matrices to represent the topological map of a graph. The continuous representation requires the definition of inner obstacles and outer boundaries.

Currently, the most common map is based on an occupancy grid map. In a grid map, an environment is discretised into a number of squares of arbitrary resolution, e.g. 1cm x 1cm, on which obstacles are marked. In a probabilistic occupancy grid, grid cells can also be marked with the probability that they contain an obstacle. This is particularly important when the position of a UGV that senses an obstacle is uncertain. Disadvantages of grid maps are their large memory requirements as well as computational time to traverse data structures with large numbers of vertices. A solution to the latter problem is the topological map that encodes entire rooms as vertices and use edges to indicate navigable connections between them.

Fig. 2.2 illustrates the representation of a grid map that describes the topology of a workspace environment based on a computer simulation. The workspace's grid is a square environment and sized in metres in a two-dimensional coordinate system, and the obstacles are occupied into different shapes and sizes. The coordinates of a starting point and a target point

have to be known based on their localisations. According to these known coordinates, a path is planned, thus, the starting and the goal points are connected through via different waypoints. The number of via points in the path is varied based on the path's length.

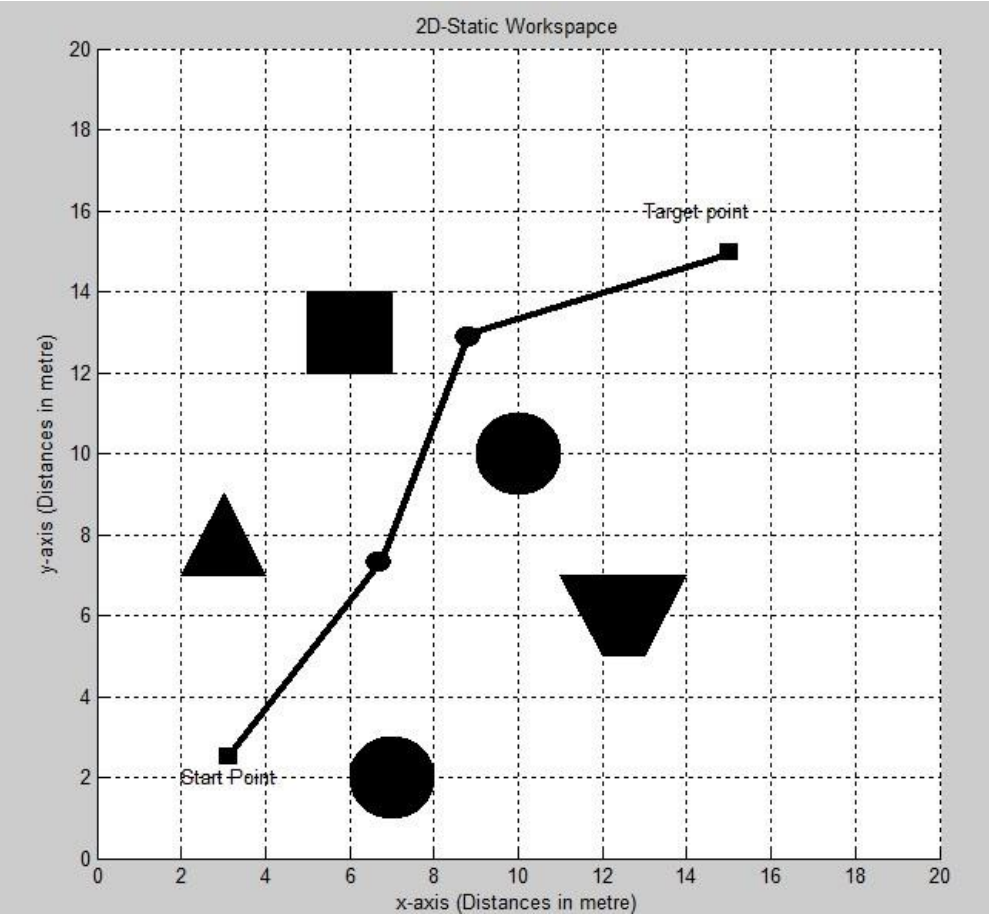



Fig. 2.2 Grid map of path planning in a 2D Environment.

### *2.3.2 Obstacle Avoidance*

Obstacle avoidance is one of the most important aspects for accomplishing the navigation task. It means that the UGV must be capable of creating a collision free path and without it movement would be strongly restrictive and fragile. Many techniques can be used for obstacle avoidance; however, the best technique depends on the topology of a specific environment. The difficulty of obstacle avoidance increases with the complexity of the structures of the specific environment. The type of workspace environments could be classified into three types; firstly, well-known environment; secondly, partially known environment and finally unknown environment. Fig. 2.3 describes a multiple of obstacles that the UGV might encounter.

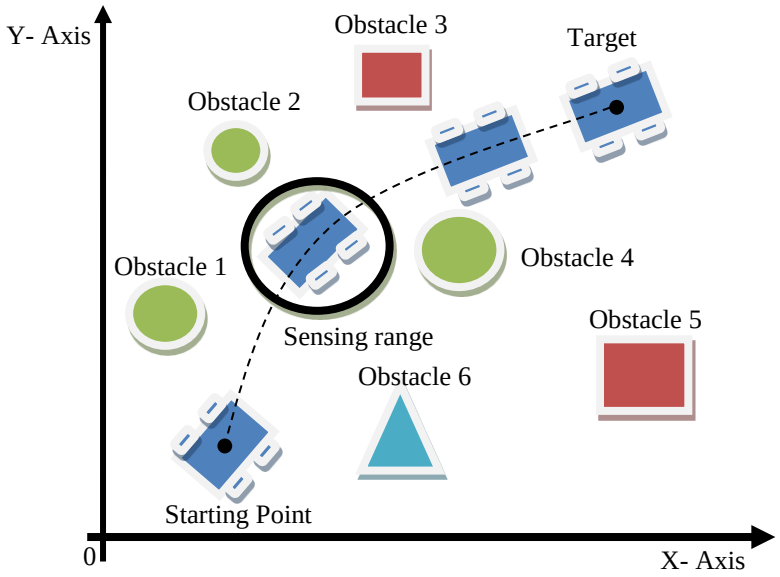



Fig. 2.3 Navigation and obstacle avoidance in a 2D coordinate system.

The collision avoidance task can be achieved accurately when the UGV takes into consideration instantaneously any obstacle within the range of surrounding's sensing of a UGV's platform. The closest obstacle has the priority to be analysed and selected for evaluation. In case of multiple obstacles are available surrounding the UGV form all the directions. Hence, the passage of the UGV might be blocked and the avoidance might not be occurred. Therefore, the best decision will be to stop the movement of the UGV until obstacles are cleared.

## 2.4 Trajectory Tracking of Unmanned Ground Vehicle

The trajectory tracking is achievable based on motion control approaches. To achieve a robust tracking, a given desired trajectory is needed to be a reference input. The control approach has to achieve a motion of a UGV in order to track a given reference and provide an actual trajectory, which should approach the reference input with a minimal trajectory tracking error. The validation criterion is based on an evaluation of a trajectory tracking error. The best trajectory tracking is when errors of both trajectory and orientation tracking are minimal.

### *2.4.1 Problem Statement*

Recently, the trajectory tracking has been a hot research area. The challenges presented by trajectory tracking come from the fact that a motion of unmanned ground vehicles in a plane possesses three degrees of freedom (DoF); whilst it has to be controlled using only two control

inputs under non-holonomic constraints. In different applications, UGVs operate autonomously over predefined trajectories to track a given trajectory in an environment. In other words, the UGVs will be enforced using a control methodology to follow the given trajectory. In the most recent studies, many algorithms and control techniques have been proposed to cope with the trajectory tracking problem. Therefore, to solve this problem, it is necessary to have a methodology that allows guiding the UGV to track a predefined trajectory. The methodology addresses the motion planning of the UGV based on the kinematic, dynamic and actuation characteristics that are taken into consideration.

### *2.4.2 Related Work*

A bibliographic review of related work that embraces different approaches of UGV trajectory tracking is provided. Padhy et al. (2010) designed a traditional proportional integral derivative (PID) controller for trajectory tracking. In spite of the structure and implementation of PID was simple and valid for tracking performance. However, the proposed controller is not sufficient for applications that require high trajectory tracking accuracy. Guo et al. (2014) reported the trajectory-tracking controller of closed-loop control structure is derived using integral back-stepping method to construct a new virtual variable. The Lyapunov theory was utilised to analyse the stability of the proposed tracking controller. Pawlowski et al. (2001) implemented a fuzzy logic for a mobile robot. The kinematic model of the mobile robot was introduced in the implementation. Antonelli et al. (2007) also proposed a fuzzy logic approach to deal with trajectory tracking problem. In this approach, the input to a fuzzy system was represented by approximate information concerning the next bend ahead the vehicle; the corresponding output is the cruise velocity that the vehicle needs to attain in order to safely drive on the path.

Shojaei e t al. (2009) presented an adaptive controller for trajectory tracking of wheeled mobile robots based on feedback linearization technique. The adaptive controller was designed based on input-output feedback linearization technique to obtain asymptotically exact cancellation for the uncertainty in the given system parameters. The presented adaptive controller was designed based on the Lyapunov approach. Keighobadi et al. (2010) designed feedback-linearization and fuzzy controllers for trajectory tracking of a wheeled mobile robot. The linguistic if-then rules of fuzzy controllers are constructed using knowledge and experience of expert humans about variations of input torque with respect to position and velocity variables.

Jiang and Nijmeijer (1997) proposed a tracking control methodology via time-varying state feedback based on the back-stepping technique. Local and global tracking problems were considered based on initial tracking error, which is set arbitrary. Hao et al. (2014) presented a trajectory tracking control methodology base on a fuzzy approach. In this methodology, both kinematic and dynamic were derived using Lagrange's equations.

Xu et al. (2014) designed fuzzy PID controller for trajectory tracking mobile robots. The controller combines between of a PID technique and fuzzy inference system. This shows a comparison between traditional PID and the integrated PID-fuzzy control. Liang et al. (2010) proposed an adaptive fuzzy control for trajectory tracking of a mobile robot. The proposed method integrated proportional derivative (PD) controller with the fuzzy controller to make use of full Benefits of both controllers. Xie et al. (2012) integrated a fuzzy control with a slide mode technique to deal with trajectory tracking problem of mobile robots. The slide mode technique implemented based on the kinematic characteristic. On the other side, the fuzzy controller used to solve the constant speed problem.

Fukao et al. (2000) integrated both kinematic controller and a torque controller for the dynamic model of a non-holonomic mobile robot. The adaptive controller for the dynamic model was designed using back-stepping method. The derivative of a torque controller was based on the kinematic controller. Solea et al. (2009) presented a slide-mode control strategy for trajectory tracking of a wheeled mobile robot. The strategy implemented in the presence uncertainties i.e. mass and moment of inertia.

Ye (2008, 2013) presented two pieces of research based on a neural network technique. The implemented architecture is based on tracking control of the velocity and orientation of a non-holonomic mobile robot. The first research is based on a PID neural network technique. This technique tracks the velocity and ordination of a non-holonomic mobile robot. The second research is a methodology based on compound sine function neural networks for tracking control of two-wheel driven mobile robot. The sine function was implemented in hidden layer based on combining a sine function with a unipolar sigmoid function. Using this method, the weight values are only adjusted between the nodes in the hidden layer and the output nodes, whilst the weight values between the input layer and the hidden layer are one, that is, constant, without the weight adjustment.

Cardenas et al.(2013) analysed the performance of a fuzzy controller and a neural network. It was implemented for an autonomous motion of mobile robots. The designed fuzzy and neural controllers were trained to optimize a given cost function by minimizing positioning error. It

was found that the mobile robot with fuzzy neural controller presents good positioning and tracking performance for different types of desired trajectories. It also showed that the fuzzy neural networks whose internal structure represents the process of fuzzy reasoning, require less training time than conventional neural networks.

The drawbacks of the aforementioned approaches are a high value of a trajectory tracking error and control efforts, especially in non-continuous gradient trajectories. Hence, the trajectory tracking error can be investigated further based on new control approaches to improve it significantly.

## 2.5 Navigation for Unmanned Ground Vehicles

The navigation of a UGV can be represented as a task of determining a collision free path that enables the UGV to travel safely between obstacles from an initial configuration to a destination configuration. The obstacle avoidance can be achieved by employing several sensors that provide information about the UGV's surroundings. Three main sensors are an ultrasound sensor, a compass sensor and an encoder sensor. The number of sensors is varied based on a design of a robotic platform. The ultrasound sensor is used to detect obstacles nearby and calculating how far are from the robotic platform. When the number of ultrasonic sensors is increased, the range of detection will be wider. The orientation of the UGV can be determined by using a compass sensor to calculate a relative angle between the UGV and a target point. The movement of the UGV can be recorded based on two-dimensional coordinates by using an encoder sensor attached to wheels.

The research communities in robotics systems have paid a great attention to develop different control architectures for aiding the navigation of UGVs. Artificial intelligence techniques and optimization algorithms have been applied widely for solving navigation problem. Artificial intelligence techniques and optimization algorithms such as neural networks fuzzy logic controllers, and genetic algorithm, particle swarm optimization, simulated annealing, ant colony optimization and artificial potential field have been widely applied for solving motion problem of UGVs and improve their performance. The ability of fuzzy logic to represent linguistic terms and reliable decision making in systems associated with uncertainty and imprecise information makes it a useful tool in control systems (Dadios, 2012). A large amount of research has been devoted based on a variety of techniques and methods, which can be summarised as in below;

### *2.5.1 Artificial Potential Field Based Navigation*

An artificial potential field (APF) method is rapidly gaining popularity for autonomous mobile robot navigations because of its simple mathematical analysis. The main idea of the potential field technique is to fill a UGV's environment with the APF where a UGV is attracted towards its goal and is repulsed from obstacles (Koren and Borenstein, 1991). Fig. 2.4 demonstrates an example of a concept of the APF. The main idea comprises of generating attraction and repulsion forces, inside a workspace environment of UGV, to guide it to a target point. The target point has an attractive force that draws the UGV. Whereas, obstacles have a repulsive force that repulses the UGV. This has been provided an elegant solution to the path-finding problem. Since a path is a result of an interaction of appropriate force fields, the path-finding problem becomes a search for optimum field configurations instead of the direct construction of an optimum path. The APF uses a vector of sums of repulsive and attractive virtual forces to compute a desired heading. The velocity of the UGV is proportional to a magnitude of a potential vector.

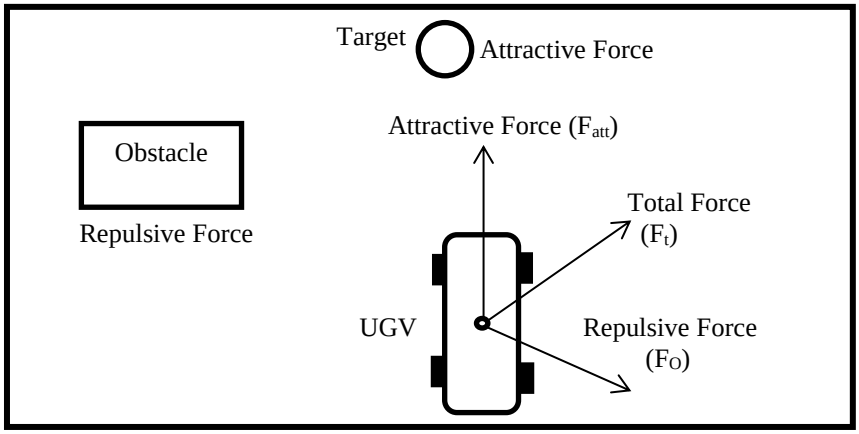



Fig. 2.4 Concept of the artificial potential field.

The APF algorithm has substantial shortcomings and these shortcomings have been identified as problems that are inherent to a principle based upon mathematical analysis. The heart of that analysis is a differential equation that combines the UGV and an environment into a unified system. In experimental work with the APF algorithm, four significant problems have been recognised that are inherent in artificial potential field methods. These inherent problems are as follows: firstly, trapping situations due to local minima, secondly, no passage between closely spaced obstacles, thirdly, oscillations in the presence of obstacles and in narrow passages (Aenugu and Woo, 2012).

Many researchers have applied this method in solving the navigation problem. Different criteria have been suggested to overcome potential filed method's drawbacks. Ge and Cui (2002) presented a new potential field method for motion planning of mobile robots in a dynamic environment where a target and obstacles are moving. The new potential functions take into account not only the relative positions of a mobile robot with respect to the target and obstacles. However, it also takes into consideration relative velocities of the mobile robot with respect to the target and obstacles. Accordingly, virtual forces are defined as a negative gradient of the potential field with respect to both position and velocity. The motion of the mobile robot is then determined by a total virtual force through the Newton's Law. Hamani and Hassam (2012) developed an APF for a navigation of a mobile robot in an unknown environment. The APF allows controlling a mobile robot to reach its goal whilst avoiding unknown obstacles on its route. The simulation results of a mobile robot are given to show the effectiveness of the proposed method.

Tang et al. (2010) presented a novel artificial potential field method for obstacle avoidance and path planning of mobile robots. An obstacle avoidance method based on gravity chain was proposed by analyzing the shortcoming of the APF method. The simulation results show that the proposed method is correct and effective. Shi et al. (2007) proposed a new method for improving artificial potential field by analysing of the disadvantage of APF such as local trapping and vibrating. By this method, a new force function was built, this in turn leads to that the trapping problem of the APF was eliminated and the vibrating problem was mitigated. In addition, the rapidness of arriving at the target was improved. The workspace of this proposed method was a static environment.

Adeli et al. (2011) introduced a new algorithm based on an artificial potential field for solving the path planning problem of mobile robots. This algorithm was built upon new potential functions based on distances from obstacles, destination point and start points. The algorithm used potential field values iteratively to find optimum points in the workspace in order to form a path from a start point to a destination point. The number of iterations depends on the size and the shape of a workspace. The performance of the proposed algorithm was tested by conducting simulation experiments. The workspace was discretised into a grid of rectangular cells where each cell was marked as an obstacle or a non-obstacle. Hence, the potential function of each cell was calculated. Simulation results showed successfully the generated paths for two workspaces.

Mohamed (2013) presented an enhanced artificial potential field planner. This planner was proposed to rapidly find online solutions for a mobile robot path planning. The required data for the algorithm are; a starting point, a target point, and readings of five infrared distance meters that were fixed in the front of the mobile robot within some angles. The classical artificial potential field represents both the repulsive force due to the detected obstacle and the attractive force due to the target. These forces can be considered as the primary direction indicator for the mobile robot. However, the classical artificial potential field has many drawbacks. Therefore, two secondary forces were suggested which are called the midpoint repulsive force and the off-sensors attractive force. These secondary forces and modified primary forces were merged to overcome the drawbacks like dead ends and U shape traps.

### *2.5.2 Fuzzy Logic Control Based Navigation*

In the past three decades, a large amount of research has been conducted based on an application of a fuzzy logic controller (FLC) for autonomous vehicle navigation. The prime target in designing the FLC for individual behaviours is to guarantee robust operation in the presence of uncertainties. The fuzzy logic controller allows to model different types of uncertainties and imprecisions; to build robust controllers starting from heuristic and qualitative models, and to integrate symbolic reasoning and numeric computation in a naturalistic framework. The FLC offers a powerful representation tool for modelling weak and imprecise information.

In literature survey, many researchers have been working on different techniques for solving path planning and navigation problems. For instance, intelligent soft computing techniques such as fuzzy control, artificial neural networks and hybrid intelligent techniques have been applied widely in solving navigation problems. Pradhan et al. (2009) presented a navigation approach for several mobile robots in an unknown environment using a fuzzy logic controller. The controller was implemented based on a different number of membership functions to navigate the mobile robots in the unstructured environment. The drawback of this approach is the representation of the mobile robots as points without considering their dimensions and dynamic characteristics.

Rashid et al. (2010) introduced an indoor navigation using a fuzzy logic controller. The controller was proposed based on the using of FLC for the target tracking control of wheeled mobile robots. The obstacle avoidance was not considered in this application. However, the controller was mainly used for the motion control. Hajar et al. (2013) presented an obstacle

avoidance approach for a mobile robot using a fuzzy logic technique. In that work, the fuzzy logic technique was constructed using eight inputs of sensory information and two outputs to control the speed and turning angle. A workspace environment was built using Webots Pro simulation software. The workspace environment consists of four static obstacles. Raguraman et al. (2009) proposed a navigation methodology for a mobile robot based on a fuzzy logic controller. The inputs and outputs of that controller were five and two respectively. The utilised environment in that methodology was a static indoor environment.

Dong et al. (2005) addressed the problem of obstacle avoidance and trajectory tracking of a desired trajectory based on a fuzzy logic controller for an unmanned aerial vehicle. The obstacles might appear unexpectedly in practical applications and might be of any shape. Simulation studies on multiple obstacles with various shapes were conducted and the effectiveness of the proposed method was verified. Islam et al., (2006) introduced a fuzzy logic algorithm for controlling an autonomous mobile robot. This algorithm enables the autonomous mobile robot to navigate in an unstructured environment. Hence, it will be capable of avoiding any obstacles might encounter its way without human intervention. The navigation task was accomplished appropriately during its reacting to avoid the crashing with obstacles by turning to a proper angle whilst moving.

Faisal et al. (2013) investigated an online navigation technique for a wheeled mobile robot in an unknown dynamic environment using fuzzy logic techniques. The aim is to apply the wheeled mobile robot for a warehouse workspace. Experimental results showed the effectiveness of the proposed algorithm. Li and Choi (2013) designed a fuzzy logic system of for path planning and obstacle avoidance in an unknown environment for a mobile robot. This fuzzy system consists of four inputs and two outputs. The obstacles in that approach were three static obstacles only. In addition, the representation of a mobile robot was represented simply as a point.

Cui et al. (2010) presented an obstacle avoidance control algorithm for a mobile robot based on a fuzzy controller. The mobile robot can detect its surrounding information by using ultrasonic sensors. The fuzzy control system was composed of dual fuzzy controllers; the first was used to control an obstacle avoiding behaviour when the mobile robot is approaching obstacles. However, the second was applied to control the mobile robot's movement to its target goal. The conducted simulation results showed that the algorithm could help the mobile robot to avoid obstacles safely. Abdessemed et al. (2014) introduced a motion control algorithm based on a fuzzy logic control for an autonomous robot navigation. In addition, a stereo vision

was applied to a path-planning module. Focusing on its kinematics characteristics and on its behavioural based fuzzy control architecture, this requires the capability to manoeuvre in a complex unknown environment. In this work, behavioural-based control architecture was adopted. This architecture was an ordered hierarchical architecture based on a fuzzy reasoning, and each local navigational task was analysed in terms of primitive behaviours.

### *2.5.3 Artificial Neural Networks Based Navigation*

Artificial neural networks (ANNs) can be widely applied to many disciplines for solving complex problems. The interests in neural networks stem from the comprehension of basic human brain functions. Mainly, artificial neural networks deal with cognitive tasks such as learning, adaptation, and optimization. Indeed, recognition, learning, decision-making and action constitute the principal navigation problems. Therefore, to solve these problems, neural networks are used. Jung et al. (1999) presented an effective method to achieve both target tracking and obstacle avoidance for a mobile robot work in an indoor environment. In addition, a wall following algorithm was employed using an artificial neural network. The ANN was trained by using back propagation method. The mobile robot was capable of reaching its destination by using the utilised tracking algorithm. The basic operation is, when the mobile robot comforts an obstacle, it avoids collision by a wall-tracking algorithm. For detecting obstacles, sonar sensors were used.

Yongjie et al. (2002) proposed a new path-planning algorithm based on neural networks for mobile robots. The neural network was used in the algorithm to model an environment and calculate a collision energy function that was the dominating term in a cost function. To implement a path-planning procedure, rather than calculating the minimum value of the cost function directly, a discrete method was used to approximate a minus gradient direction of the cost function in order to determine a motion tendency of a set of point alongside a path. Janglová (2004) solved a motion-planning problem for mobile robot control using neural networks. In this method, a collision-free path was constructed to achieve the movement of a mobile robot among obstacles based on two neural networks. The first neural network was used to determine a free space using data from ultrasonic range finders. The second neural network was based on finding a safe direction for the next position of a path in an environment whilst avoiding the nearest obstacles. Arbitrary shapes and sizes of obstacles were involved in this method.

Engedy and Horváth (2010) introduced an artificial neural network based on a motion control and a path planning system of a wheeled mobile robot navigating among multiple obstacles. The presented neural network considers distance sensor readings and relative position of a target point. The neural network was used was trained using back propagation optimisation algorithm. In addition, it employs potential fields for obstacle avoidance purposes. For a construction of a path, a numerous of waypoints might be combined to create the path. The mobile robot was able to follow the path, without colliding with any obstacle. Velagic et al. (2010) implemented a neural network controller to control the velocity for a kinematic model of a mobile robot. The controller was trained online by using back propagation algorithm to track a predefined path. In such a case, it would be useful to set a number of waypoints to be followed by the mobile robot and to avoid obstacles.

Chi and Lee (2011) proposed a neural network control system that is able to guide mobile robots traverse through a maze with arbitrary obstacles. The patterns were trained by using Matlab toolbox for a motion control. There were many specific patterns which defined to help the mobile robot to organize its situation. Sonar and laser range finders are two main sensors for passing on information about the environment. The neural network approach had demonstrated the effectiveness on avoiding obstacles and the mobile robot can navigate through the trajectory with a stability and reliability. Zhao and Wang (2012) introduced an autonomous navigation system based on neural networks for mobile robots. In this research, a navigation system was developed with a learning capability to adapt to an unknown environment. Sonar sensors were incorporated by the neural network to solve the problem of an autonomous robot navigation.

Based on the literature, the most of work has been conducted in static environments. This represents one of the main drawbacks in addition to the previous problems.

### *2.5.4 Evolutionary and Swarm Intelligence Algorithms Based Navigation*

Evolutionary and swarm intelligence algorithms have been approved to be beneficial for tackling challenging of search and optimization problems. They are used to find the best trajectories for a certain task. If an unmanned ground vehicle works in a manufacturing plant, the evolutionary and swarm intelligence algorithms could figure out the best solution for such problem (Simon, 2013). Miao and Tian (2008) proposed a simulated annealing algorithm to determine an optimal approach or a near optimal path for a mobile robot in dynamic environments that filled with static and dynamic obstacles. The approach had used vertices of

obstacles to define a search space. It processes off-line computation based on known static obstacles, and then, re-computes a route online, if a moving obstacle was detected.

An obvious example of evolutionary algorithms is a genetic algorithm (GA). The principle operation of GA consists of a set of solutions, which could be represented by chromosomes. The solutions are called a population. Based on solutions from one population, a new population will be formed and generated for a new generation. This is achieved with the hope that the new generation will be better than the old one. Solutions that are selected to form new solutions called an offspring. The offspring is selected based on a fitness function; the more suitable offspring has more chances to be reproduced. In additional, the process of GA is based on deploying different techniques to obtain the best solution such as coding, crossover and mutation. This process is repeated until some predefined condition is satisfied or the best solution is achieved (Uszkoreit et al., 2007). In additional, GA is a specifically useful tool when a problem under consideration does not have an accurate mathematical representation. It has been established that is an advantageous tool for a robotic navigation. Moreover, It has been utilized in a motion planning problem in different environments (Tewolde, 2013).

Elshamli et al. (2004) introduced a genetic algorithm path planner for solving the path-planning problem in stochastic environments. The genetic algorithm path planner was based on a variable length of chromosomes for path encoding, where different evolutionary operators are applied. A generic fitness function was used to combine all the objectives of the problem. The paths are evolved using specially designed random operators and specific domain knowledge operators. Davies and Jnifene (2006) utilised a genetic algorithm that was capable of generating an optimal (shortest distance) path plan for a mobile robot to visit all of e specified waypoints without colliding with known obstacles. Once a path plan had been determined, a suitable trajectory-planning program needs to provide required velocities and accelerations for a mobile robot to reach each waypoint. It was shown that the choice of search parameters for the genetic algorithm effected execution time of searching for a solution. The genetic algorithm path planner then successfully guided an actual the mobile robot to its waypoints without colliding with obstacles surrounding its platform.

Ghorbani et al. (2009), a global path planning is considered based on genetic algorithm to reach an optimum path for a mobile robot with obstacle avoidance. Two-dimensional coding for a path via points was converted to one-dimensional coding for reducing complexity. The parameters of collision avoidance and the shortest distance were integrated into a fitness function. The simulation results showed that the proposed method is an accurate and effective.

Shi and Cui (2010) presented a dynamic path-planning scheme based on a genetic algorithm for a navigation and an obstacle avoidance of a mobile robot in an unknown environment. The fitness function of the genetic algorithm takes a full consideration of three factors; the collision avoidance, the shortest distance of a path and smoothness of the path. The simulation results verify that the genetic algorithm is an effective approach.

Yun et al. (2010) proposed an improved genetic algorithm of an optimum path planning for a mobile robot navigation. An obstacle avoidance algorithm and a distinguish algorithm were introduced to generate an initial population and check paths in order to come out with all feasible individuals in a first generation. This technique was used to avoid all types of obstacles that are detected by sonar sensors of a mobile robot in an environment. The feasibility of the proposed genetic algorithm had been verified and proven by testing the algorithm in a workspace filled with obstacles located randomly. Mohanta et al. (2011) presented a novel knowledge based on a genetic algorithm for a path planning of multiple robots to reach multiple targets based seeking behaviour in the presence of obstacles. The genetic algorithm had been incorporated in the Petri-Net model to make an integrated navigational controller. The proposed algorithm was based upon an iterative non-linear search, which utilises matches between an observed geometry of an environment and a priori map of position locations, to estimate a suitable heading angle, thereby correcting a position and an orientation of the mobile robots to find targets. This knowledge based GA was capable of finding an optimal or near optimal robot path in complex environments.

Tuncer and Yildirim (2012) introduced a new mutation operator for a genetic algorithm and applied it to a path-planning problem of mobile robots in dynamic environments. The improved mutation method simultaneously checks free nodes close to mutation node instead of randomly selecting a node one by one. The method accepts the node according to a fitness value of a total path instead of a direction of movement through a mutated node. Whereas, a conventional random mutation operator in a simple genetic algorithm or some other improved mutation operators might cause infeasible paths. Achour and Chaalal (2011) investigated the application of a genetic algorithm for solving the problem of a path planning for mobile robots. In this approach, a population of paths was obtained using a random distribution strategy. The performance of a proposed genetic algorithm was examined in complex environments. The obtained results showed that this approach can find an optimal path in a short time and has the capacity to enrich the configuration space by a different set of eligible movements based on a crossover operator and selection. Zhao and Gu (2013) utilised an environment model based on

grid for performing a path-planning task. A two-layer genetic algorithm mechanism was used as a searching tool to find an optimal path in a given environment. Each layer has different fitness function. The first layer was employed for avoiding of static obstacles and the second layer was engaged for avoiding of dynamic obstacles. In which case study, only one moving obstacle are considered with multiple static obstacles.

Raja and Pugazhenthi (2009) presented a particle swarm optimisation algorithm to determine an optimal planning solution for a mobile robot in dynamic environments. Obstacles of different shapes with varying velocities were considered. The generated valid paths were subjected to the particle swarm optimisation to acquire global optimal paths. The proposed algorithm also gives the velocity of the robot for each path segment depending upon optimisation of the path length and elapsed time. The effectiveness and efficiency of the proposed algorithm is demonstrated by simulation studies. Zhang et al. (2013) proposed a multi-objective path-planning algorithm based on particle swarm optimization for a robot navigation in a workspace. Several new operations were incorporated into the proposed algorithm to improve its effectiveness, such as the particle updating method based on random sampling and uniform mutation. The simulation results demonstrated the capability of the proposed method to generate optimal paths.

Cong and Ponnambalam (2009) proposed an ant colony optimization (ACO) to solve a mobile robot path-planning problem. Several maps of varying complexity were used in order to demonstrate the effectiveness of ACO algorithms in solving the path-planning problem. Each map consists of static obstacles and walls in different arrangements. The ants, which representing the mobile robots were placed at different starting points. The ants would then have to find their way towards destinations whilst avoiding all obstacles and walls approaching its way. Lee et al. (2011) presented a novel heterogeneous ant colony optimization (HACO) algorithm to solve a global path-planning problem. The performance of HACO algorithm was compared with a modified genetic algorithm for global path planning. The simulation results demonstrated that the proposed HACO algorithm provides a better performance than a conventional genetic algorithm in overall.

Although some of the conducted optimisation algorithms have been tackled a path planning and obstacle avoidance in dynamic environment, some drawbacks have been noticed in the conduced work in the literature. Most methodologies were based on a global path planning and this acquires a prior knowledge of environments. Hence, a reactive navigation is needed to deal with randomly changing environments. Autonomous platforms were represented as points and

this facilitates navigation missions significantly to pass through narrow passages. Consequently, such limitations are necessary to be considered in order to establish a robust navigation methodology.

### *2.5.5 Hybrid Intelligence Techniques Based Navigation*

In the previous sections, standalone techniques have been used to deal with a navigation problem such as potential filed, evolutionary and swarm intelligence algorithms, artificial neural networks and fuzzy logic techniques. These techniques are not always the best choice for some tasks that can be effectively produced a robust performance in dynamic and complex environments. The combination of two techniques or more might be involved to obtain better performance. Du et al. (2005) introduced a global path planning based on both a neural network and a genetic algorithm. The neural network model was constructed from information in an environment of a mobile robot. Then, the model was used to establish a relationship between a collision avoidance path and the output of the model. Hence, a two-dimensional coding for the path via-points was converted to one-dimensional. The genetic algorithm was applied to find the global optimal path in a static workspace.

Parhi (2008) discussed the application of a neuro-fuzzy controller for a task of navigation of multiple mobile robots. In the controller, the output of an artificial neural network was fed as an input to a fuzzy controller. Hence, the final outputs from the fuzzy controller were used for a motion control of the multiple mobile robots. The inputs to the neural network were obtained from mobile robots' sensors (such as left, front, right obstacle distances and the target angle). The implemented neural network consists of four layers and the back propagation algorithm was used to train the neural network. The output from the neural network was an initial steering angle. The output of the ANN accompanied with left, front, right obstacle distances represent the inputs to the fuzzy controller. The outputs from the fuzzy controller were the crisps values of left and right wheel velocities. From the left and right wheel velocities, the final steering angle of the mobile robots was calculated. He et al. (2008) proposed a fuzzy neural network method based on Takagi-Sugeno model. The navigation algorithm based on Takagi-Sugeno model was carried out by utilizing a collection of data from eight ultrasonic sensors. The test results showed that a mobile robot using this fuzzy neural network could detect obstacles in different workspaces.

Ganapathy et al. (2009) utilised a fuzzy logic and an artificial neural network to help an autonomous mobile robot of moving, learning its environment and reaching the destination.

The study was conducted based on four combinations of training algorithms, which were composed of the fuzzy logic and the artificial neural network for avoiding acute obstacles in a given environment. In addition, a path-remembering algorithm was proposed to assist a mobile robot to come out from acute obstacles. Moreover, a virtual wall creation method was presented in order to prohibit the mobile robot from re-entering the same acute obstacle once it had been turned away from the wall. A goal seeking was the main behaviour that controls the mobile robot to reach its target. The mobile robot was capable of reaching the desired goal even when the environment consists of some odd shaped acute obstacles such as 'U' or 'V' shaped ones. Joshi and Zaveri (2010) developed a neuro-fuzzy system for reactive behaviour based navigation of a mobile robot. In the system, wheels' velocities were controlled by based on sensory information. The mobile robot performed the reactive navigation rather than looking for optimal path as performed by path planning techniques. The performance of a neuro-fuzzy system was compared to neural and fuzzy approaches.

Vukosavljev et al. (2011) used a combination of two navigation algorithms; firstly, a self-learning neural network was used to form a movement plan for a mobile robot; secondly, a collision-free control algorithm based on heuristic neuro-fuzzy approach was presented. The basic task of a neural network was to generate an initial path. Both algorithms were adapted and implemented to navigate a platform of a mobile robot equipped by two independent wheel drives, encoders and a set of short-range sonars. The combined reactive algorithm was used in real time for accomplishing the navigation mission of a robotic system. The navigation algorithms were placed into a personal computer, which was connected to the mobile robot wirelessly and based on a wired link as well.

Khelchandra et al. (2014) presented a technique of solving a motion-planning problem of a mobile robot using artificial neural network, fuzzy logic and genetic algorithm. In this technique, a path from a set of numerous paths was selected by using trained artificial neural networks for the mobile robot to be moving straight towards a target point. A fuzzy logic was used to avoid collision when obstacles obscure all the paths. Genetic algorithm was applied as an optimizer to find optimal locations along the obstacle free directions and positions by choosing a set of fuzzy rules for the fuzzy logic system from a large rule base engine. The obtained results showed that a combination of those features was computational efficient by helping each other to eliminate their individual limitations.

Deshpande and Bhosale (2013) introduced an adaptive neuro-fuzzy inference system (ANFIS) technique to solve a navigation problem of a mobile robot. The sensory information

is based on magnetic compass and sonar, which they represent the inputs to the ANFIS network, the output of this network is motor velocity to control the motion of the robot. The path planning in this technique was examined on a simple static environment consists of three obstructing obstacles. Notwithstanding, the obtained path is not completely feasible and relatively long.

In addition to the aforementioned drawbacks, other limitations can be considered to improve a navigation performance based on real world scenarios. For example, kinematics and dynamics constraints have not been widely applied to investigate its influence on the motion control. Moreover, considering an occupied size of an unmanned ground vehicle and obstacles are essential to demonstrate the vigorous and effectiveness of proposed techniques.

## 2.6 Chapter Summary

In this chapter, the problem of trajectory tracking is intensively reviewed based on the state of the art. It has been noticed that there are still some gaps to improve the tracking response further and minimise the tracking error. It is observable that some techniques such as a fractional order proportional integral derivative controller has not been utilised yet to investigate its performance with such a problem in unmanned ground vehicles. Such controllers can be also integrated with other techniques to create new control approaches to enhance its effectiveness i.e. a neural network to create fractional order proportional integral derivative-neural network, FOPID-NN, controller. Secondly, the navigation problem is intensively studied based on a variety of control methodologies. It is noticeable that most the conducted work is based on static environments. Even though, it was assumed that the locations of obstacles were already known. Thus, this considers as a global navigation, which it would be easier to be handled. In addition, despite of some works are also conducted in dynamic environments. However, the movement of obstacles was invariant. The motion of a UGV was mainly assumed as a point. Hence, this simplifies the navigation, especially in narrow passages. Hence, the dimensions of the UGV are important to be considered based on real world applications. Random movement of dynamic obstacles will be utilised on our proposed workspaces. Fuzzy inference systems and adaptive neuro-fuzzy inference systems will be introduced to tackle the navigation based on a reactive approach where the UGV does not have prior knowledge about its operating environment.

# Chapter 3

# Modelling of UGV and Trajectory Tracking Based on PID Controller

## 3.1 Introduction

The mathematical modelling is a key procedure for representing and analysing the behaviour of a system. It is required for designing and analysing of control systems. In addition, it is an essential step for the development and controllability of optimal systems. An accurate representation of some models might be a challenge based on several facts i.e. linear and nonlinear systems. For instance, the modelling of a system is increased in terms of complexity if the system is nonlinear and especially with higher order systems. Some linearization approaches might be utilised to overcome complexity of modelling. However, this might have an influence over the accuracy of the system. Based on the advances in modelling techniques, many approaches have been proposed for facilitating the modelling processes such as a database model that involves creating some input data to be fed into a system, and based on outputs, the system modelling and can be identified.

Robotic systems generally consist of different components of subsystems such as electrical, electronic and mechanical subsystems. In other words, it can be defined in term of hybrid systems. The modelling process of each component of the robotic systems involves analogous procedures for each component. The components of robotic systems in somehow, they are linked and integrated with each other based on the defining inputs and outputs of each component. This in turn will lead for constructing a model of each subsystem that can be combined to obtain the overall model. The modelling of an unmanned ground vehicle platform consists of kinematic, dynamic and electrical actuating units that model the driving of the whole platform. Kinematic modelling deals with the geometric relationships that govern the system and studies the mathematics of motion without considering affecting forces. On the other hand, dynamic modelling studies the motion in which forces and energies take into consideration as part of the model. The kinematic and dynamic modelling is considered as the mechanical part of the robotic system. Actuator modelling is needed to find the relationship between driving control signals and inputs of a mechanical system.

The modelling of each part of the robotic system is conducted and discussed separately throughout this chapter. The simulation of the implemented model will be accomplished and

demonstrated after obtaining all equations that govern the system entirely. The simulation process based on four main trajectories is investigated to examine the validation the model using an appropriate computer package. The first step for the mechanical modelling is to define an appropriate coordinate system for the platform, which is described in the next section.

The implementation of the model is based on MATLAB-Simulink software package. The MATLAB-Simulink is an important research tool for performing a sophisticated modelling and it is a powerful tool for a graphical visualization, planning, and strategic development in different areas of research. It provides a huge range of built in functions and tools for accomplishing the modelling accurately. In addition, it facilitates the programming of coding for users for innovation and creativity based on new applications. For instance, mathematical equations that describe the behaviour of a model can be coded based on a user experience and its proficient. One of the more convenient approaches that MATLAB provides is a state space modelling. This is based on defining inputs and outputs of a system and then creates a matrix of parameters that manipulates inputs with outputs of the modelled system.

## 3.2 Chapter Organisation

The chapter is organised as follows: In the next section, locomotion of the UGV is discussed based on two categories i.e. holonomic and non-holonomic. Section 3.4 is dedicated to achieve the modelling of the UGV, it comprises kinematic and dynamic analysis in addition to the actuation unit. The designing of a traditional PID controller is described in Section 3.5. It has been devoted to solve the trajectory tracking problem. In Section 3.6, the simulation results are conducted based on four different scenarios to investigate the performance of the traditional PID controller. Finally, the chapter summary is introduced in Section 3.7.

## 3.3 Locomotion of Unmanned Ground Vehicle

The locomotion is the power of movement of an unmanned ground vehicle (UGV) from one place to another. Hence, locomotion is concerned with interaction forces, mechanism and actuators that generate those forces. Locomotion and mobility of the UGV can be depended on several facts such as a number of contact points or areas, an angle of contact, a friction, a centre of gravity, an inclination of terrain, an environment structure and a medium of environment (water, air, soft or hard ground). Recent advances in robotic technology have made some of robotic platforms are capable of working on different medium such as land, air, water and space.

The locomotion understanding of UGV implies finding out how wheels are constrained to affect the motion of the UGV. The motion of the UGV can be analysed based on a structure of wheels and how they can be driven. The motion of a UGV can be constrained based on two mechanisms, i.e. holonomic and non-holonomic constraints. These two terminologies will be explained in the following subsections.

### *3.3.1 Holonomic Unmanned Ground Vehicle*

When kinematic constraints can be integrated to yield constraints on position variables, these are called holonomic constraints. Additionally, this refers to the relationship between controllable and total degrees of freedom of the UGV. If the controllable degree of freedom is equal to total degrees of freedom, then, the UGV is holonomic. When the UGV is built in castor wheels or Omni-wheels that is an explicit example of the holonomic drive as it can freely move in any direction and the controllable degrees of freedom is equal to total degrees of freedom. Figs. 3.1(a) and 3.1(b) demonstrate a castor wheel and an Omni-wheel which they can rotate in both X-axis and Y-axis making it move in both the directions (Holmberg and Khatib, 1999).

(a) (b)

Fig. 3.1 Holonomic wheels a) Castor wheel b) Omni-wheel.

### *3.3.2 Non-Holonomic Unmanned Ground Vehicle*

When kinematic constraints for which an integration is not possible, called non-holonomic constraints. A typical example of a non-holonomic constraint is a wheel rolling vertically without slipping on a surface. The constraint on an allowable velocity (the point of contact of the wheel with the surface cannot slip in all directions) cannot be integrated to yield a constraint on the position of the wheel. Furthermore, if the number of controllable degree of freedom is less than the total degrees of freedom, then it is known as non-holonomic drive. For instance, when a vehicle has three degrees of freedom; i.e. its position in two axes and its orientation. However, there are only two controllable degrees of freedom that are accelerating or braking and turning angle of steering wheels. This makes it difficult for a driver to turn a vehicle in any direction unless it skids. The vehicle will move in Y-axis forward and backward and will never

make any movement in the X-axis (Barraquand and Latombe, 1989). Fig. 3.2 demonstrates a typical example of non-holonomic unmanned ground vehicles.

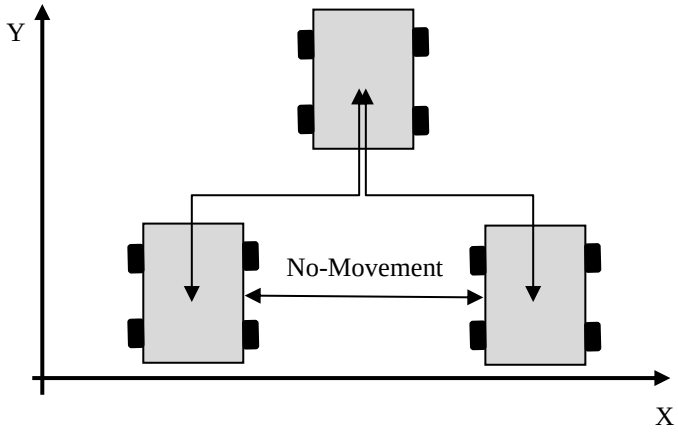



Fig. 3.2 Non-holonomic of unmanned ground vehicles.

In addition, based on which approach that utilises to drive a UGV, the driving mechanism can be classified into four main types of driving systems as follows:

***1) Differential Drive***

Many unmanned ground vehicles use a driving mechanism known as differential drive. It consists of two driving sides mounted on a common axis, and each side of a vehicle can have either one or two wheels. However, whatever each side has of wheels, they will be driven independently and being driven either forward or backward. Whilst the velocity of each side can be varied, the vehicle must rotate about a point that lies along their common left and right wheels' axis. The point that the UGV rotates about is known as the instantaneous centre of curvature (ICC) as shown in Fig. 3.3. By varying the velocities of two wheels, the motion can be varied to track given trajectories.

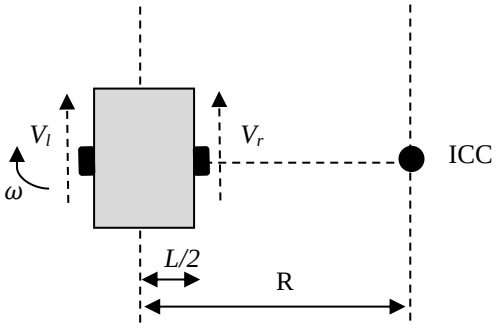



Fig. 3.3 Differential drive vehicle.

$$\omega\left(R+\frac{L}{2}\right)=V_r \tag{3.1}$$

$$\omega\left(R-\frac{L}{2}\right)=V_l \tag{3.2}$$

where $L$ is the distance between the centre of the two wheels, $V_r$, $V_l$ are the right and left wheel velocities along the ground, and $R$ is the signed distance from the ICC to the midpoint between the wheels. At any instance of time, Both $R$ and $\omega$ can be calculated as follows:

$$R=\frac{L}{2}\left(\frac{V_l+V_r}{V_r-V_l}\right) \tag{3.3}$$

$$\omega=\frac{V_r-V_l}{L} \tag{3.4}$$

***2) Synchronous Drive***

In a synchronous drive, each wheel is capable of being driven and steered separately. However, a typical configuration of wheels allows all wheels to turn and drive in unison, this leads to a holonomic behaviour. The orientation of a steered wheel of the rotation axis can be controlled as shown in Fig. 3.4. In addition, the three wheels point out in the same direction and turn at the same rate. This is typically achieved with a complex collection of belts that physically links the wheels together. The vehicle controls the direction the wheels steering and the rate at which they roll. Because of all wheels remain parallel, the synchronous drive always rotates about the centre of the UGV. The synchronous drive has the ability to control the orientation of their pose directly.

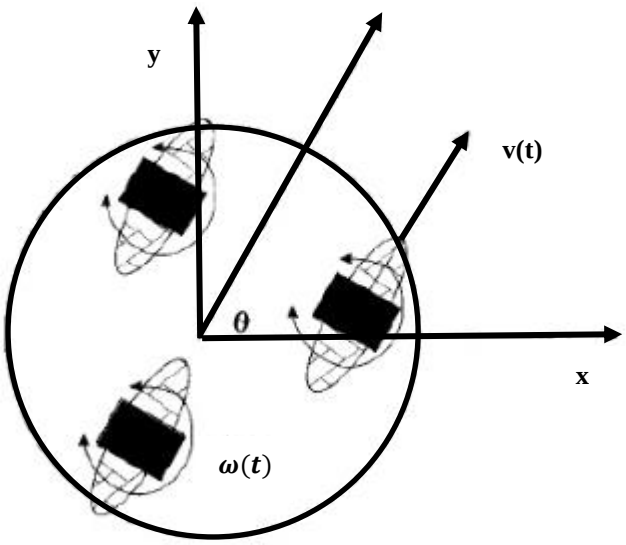



Fig. 3.4 Synchronous drive vehicle.

### *3) Car Drive (Ackerman Steering)*

It is an elegant and a simple mechanism to approximate ideal steering used in motor vehicles, the inside front wheel is rotated slightly sharper than the outside wheel to reduce tire slippage. Ackerman steering provides nearly an accurate dead reckoning solution whilst supporting traction and ground clearance. Generally, the method of choice for outdoor autonomous vehicles. An example of an Ackerman steering drive vehicle is shown in Fig. 3.5 below.

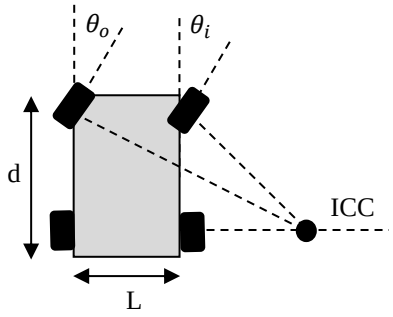



Fig. 3.5 Ackerman steering drive vehicle.

The Ackerman steering equation:

$$\cot\theta_i - \cot\theta_o = \frac{d}{L} \tag{3.5}$$

where,

$d$ = lateral wheel separation,

$L$ = longitudinal wheel separation,

$\theta_i$ = relative angle of inside wheel,

$\theta_o$ = relative angle of outside wheel.

### *4) Omni-Directional Drive*

The term of omni-directional is used to describe the ability of a system to move instantaneously in any direction from any configuration. When a vehicle has holonomic constraints, it can travel in every direction under any orientation. This capability is widely known as omnidirectional mobility. Omnidirectional vehicles have great advantages over conventional non-holonomic platforms, with car-like Ackerman steering or differential drive system, for moving in tight areas (Borenstein et al., 1997). They can crab sideways, turn on the spot, and follow complex trajectories. Based on Omni-directional mechanism, UGVs are capable of easily performing tasks in environments with static and dynamic obstacles and narrow aisles. Such environments are commonly found in factory workshop offices,

warehouses and so on. In contrast, non-holonomic UGVs can move in forward and backword directions and describe some curved trajectories. However, they cannot crab sideways. For example, for parallel parking, a differential drive vehicle should make a series of manoeuvres (Doroftei et al, 2007). Fig. 3.6 introduces an example of an Omni-directional drive vehicle.

Fig. 3.6 Omni-directional drive vehicle.

## 3.4 Modelling of Unmanned Ground Vehicle

The modelling of an unmanned ground vehicle is analysed and described in the following sections. This modelling includes analysis of the kinematic, dynamic and actuating characteristics of the UGV. The kinematic model describes the motion of the vehicle without considering the forces cause this motion. The dynamic model takes into consideration the forces and torques that cause the motion. Finally, the actuator modelling provides the analysis of power sources that generate the torques.

### *3.4.1 Kinematic Modelling*

The kinematic model of an unmanned ground vehicle in a two-dimensional plane can be conducted by using Cartesian coordinates. It is assumed that the UGV moves without slipping on a plane, that means there is a pure rolling contact between the wheels and the ground and there is no lateral slip between the wheel and the plane. The vehicle has four fixed standard wheels and is differentially driven by skid steering motion. The two wheels on front side are driven simultaneously with the two wheels on the rear side. The wheels have the same radius '$r$'. The driving wheels are separated by distance '$L$'. The position of the vehicle in the two dimensional plane at any instant is defined by the situation in Cartesian coordinates and the heading with respect to a global frame. The configuration of the UGV is represented by generalized coordinates, $P_c = (X_c, Y_c, \theta)$. The schematic diagram of the unmanned ground vehicle is depicted in Fig. 3.7.

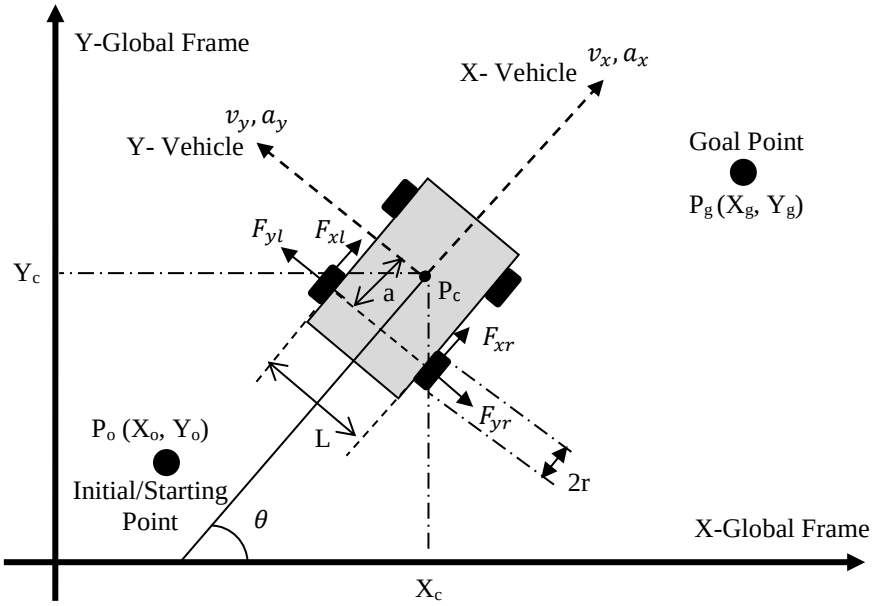



Fig. 3.7 Schematic diagram of the unmanned ground vehicle .

A set of relationships for the unmanned ground vehicle can be defined as:

$$\begin{bmatrix} V_x \\ V_y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ -\frac{r}{L} & \frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} \quad (3.6)$$

$$v = r.\left[\frac{\omega_r + \omega_l}{2}\right] \quad (3.7)$$

$$\dot{\theta} = r.\left[\frac{\omega_r - \omega_l}{L}\right] \quad (3.8)$$

$$\dot{x}_c = v\cos\theta \quad (3.9)$$

$$\dot{y}_c = v\sin\theta \quad (3.10)$$

$$x_c = S\cos\theta \quad (3.11)$$

$$y_c = S\sin\theta \quad (3.12)$$

$$S = \frac{S_r + S_l}{2} \quad (3.13)$$

$$\theta = \frac{S_r - S_l}{L} \quad (3.14)$$

$$\omega = \dot{\theta} \quad (3.15)$$

$$\omega_r = \dot{\emptyset}_r \quad (3.16)$$

$$\omega_l = \dot{\emptyset}_l \quad (3.17)$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ Sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} \quad (3.18)$$

In addition, it is assumed that the UGV is subject to the kinematic constraints such as the contact between the wheels and the ground is pure rolling, and non-slipping (Fierro and Lewis, 1998).

**1) No slip constraint**

$$\dot{y_c}\cos\theta - \dot{x_c}\sin\theta = a\,\dot{\theta} \tag{3.19}$$

**2) Pure rolling constraint**

$$\dot{x_c}\cos\theta + \dot{y_c}\sin\theta + L\,\dot{\theta} = r\,\dot{\emptyset}_r \tag{3.20}$$

$$\dot{x_c}\cos\theta + \dot{y_c}\sin\theta - L\,\dot{\theta} = r\,\dot{\emptyset}_l \tag{3.21}$$

These constraints demonstrate that the driving wheels do not slip. The three non-holonomic constraints can be written in the following form:

$$A(q)\dot{q} = 0 \tag{3.22}$$

$$A(q) = \begin{bmatrix} -sin\theta & cos\theta & a & 0 & 0 \\ cos\theta & sin\theta & L & -r & 0 \\ cos\theta & sin\theta & -L & 0 & -r \end{bmatrix} \tag{3.23}$$

$$\dot{q} = [\dot{x_c} \quad \dot{y_c} \quad \dot{\theta} \quad \dot{\emptyset}_r \quad \dot{\emptyset}_l\,]^T \tag{3.24}$$

The above system can be transformed into a more proper representation for control and simulation purposes. In this transformation, it is required to find a method to eliminate the constraint term from the equation. The kinematic matrix is defined by the following transformation:

$$\dot{q} = \begin{bmatrix} \dot{x_c} \\ \dot{y_c} \\ \dot{\theta} \\ \dot{\emptyset}_r \\ \dot{\emptyset}_l \end{bmatrix} = \begin{bmatrix} cos\theta & -asin\theta \\ Sin\theta & acos\theta \\ 0 & 1 \\ \frac{1}{r} & \frac{L}{r} \\ \frac{1}{r} & -\frac{L}{r} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.25}$$

This model is referred to a vehicle kinematic model since it describes velocities but not forces that have an effect on the velocity. The wheels' radius and the distance between driving wheels is assumed 'r=0.06m' and 'L=0.20m', respectively. The distance between the centre of mass and the rear driving wheels 'h' equals '0.10 m'. In the next section, the dynamic model will be analysed.

### *3.4.2 Dynamic Modelling*

The dynamics model of an unmanned ground vehicle represents the study of the relationship between the various forces action on a robotic mechanism and their accelerations. This is mainly used for simulation study and analysis of the vehicle's design and a motion controller design for the vehicle. The description of the mechanism of the robot movement is given in terms of its component parts; bodies, joints and the parameters that characterise them. In fact, several parameters are required to define the dynamic model of a given rigid body such inertia, centre of mass and applied forces. The Newton-Euler approach can be used to derive the dynamic model of the unmanned ground vehicle which is represented in the following general form (Fierro and Lewis, 1997; Mohareri, 2009):

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(q,\dot{q}) + G(q) = B(q)T \tag{3.26}$$

where,

$q \in \mathcal{R}^{nx1}$ is a vector of generalized position coordinates

$\dot{q} \in \mathcal{R}^{nx1}$a vector of longitudinal and angular velocities of generalized coordinates.

$M(q) \in \mathcal{R}^{nxn}$ is the symmetric positive definite inertia matrix of the system,

$C(q,\dot{q}) \in \mathcal{R}^{nx1}$ is the centripetal and Coriolis forces matrix,

$F(\dot{q})$ is the surface friction matrix,

$G(q)$ is the gravitational vector,

$B(q) \in \mathcal{R}^{nx(n-m)}$ is the input transformation matrix,

$T \in \mathcal{R}^{(n-m)x1}$ is the input transformation matrix and input vector.

The vehicle planar motion leads to the elimination of the gravity terms in the dynamic equation:

$$G(q)\ \&\ F(q,\dot{q}) = 0 \tag{3.27}$$

Therefore, Eq. (3.26) can be rewritten in another appropriate manner as follows:

$$\overline{M}(q)\dot{\omega} + \overline{C}(q,\dot{q})\omega = B(q)T \tag{3.28}$$

It is observable that the only forces acting on the vehicle are actuator forces acting on left and right wheels as shown in Fig 3.7 given previously. The model derivation can be started by representing the vehicle position using polar coordinates.

Assuming that the vehicle is a rigid body, its position can be represented using its angle and radius:

$$\vec{r} = re^{i\theta} \tag{3.29}$$

By differentiating the position vector, the velocity and acceleration of the vehicle can be calculated:

$$\dot{\vec{r}} = \dot{r}e^{i\theta} + r\dot{\theta}ie^{i\theta} \tag{3.30}$$

$$\ddot{\vec{r}} = \ddot{r}e^{i\theta} + \dot{r}\dot{\theta}ie^{i\theta} - r\dot{\theta}^2e^{i\theta} + \dot{r}\dot{\theta}ie^{i\theta} + r\ddot{\theta}ie^{i\theta} \tag{3.31}$$

These two above equations can be simplified and re-written in radial and tangential terms as follows:

$$\dot{\vec{r}} = \dot{r}e^{i\theta} + r\dot{\theta}e^{(i\theta+\frac{\pi}{2})} \tag{3.32}$$

$$\ddot{\vec{r}} = (\ddot{r} - r\dot{\theta}^2)e^{i\theta} + (2\dot{r}\dot{\theta}e^{i\theta} + r\ddot{\theta})e^{(i\theta+\frac{\pi}{2})} \tag{3.33}$$

The radial and tangential velocity and acceleration terms are defined as follows:

$$v_x = \dot{r} \tag{3.33}$$

$$v_y = r\dot{\theta} \tag{3.34}$$

$$a_x = \ddot{r} - r\dot{\theta}^2 \tag{3.35}$$

$$a_y = 2\dot{r}\dot{\theta} + r\ddot{\theta} \tag{3.36}$$

From the above four equations, we can write the following relations between the radial and tangential velocity and acceleration of the robot:

$$a_x = \dot{v}_x - v_y\dot{\theta} \tag{3.37}$$

$$a_y = \dot{v}_y - v_x\dot{\theta} \tag{3.38}$$

It is needed to write Newton's second law in radial (x) and tangential (y) directions to find the relation between the forces and accelerations:

$$\sum F_x = ma_x$$

$$ma_x = F_{xl} + F_{xr} \tag{3.39}$$

$$\sum F_y = ma_y$$

$$ma_y = F_{yl} + F_{yr} \tag{3.40}$$

The acceleration terms from Equations (3.37) and (3.38) are substituted in Equations (3.39) and (3.40), that yields the acceleration of the vehicle in terms of the actuating forces and the velocity terms as shown in the following two equations:

$$\dot{v}_x = v_y\dot{\theta} + \frac{F_{xl} + F_{xr}}{m} \tag{3.41}$$

$$\dot{v}_y = -u_x\dot{\theta} + \frac{F_{yl} + F_{yr}}{m} \tag{3.42}$$

The Newton's second law in rotation about the centre of mass can be calculated as follows:

$$\sum M_c = I_c\ddot{\theta}$$

$$\ddot{\theta} = -\frac{(F_{yl} + F_{yr})a}{I_c} + \frac{(F_{xr} - F_{xl})L}{I_c} \tag{3.43}$$

Based on the assumptions that no sliding on lateral direction and pure rolling on longitudinal direction, the lateral velocity of the midpoint if the drive wheels is zero. Therefore, the lateral velocity and acceleration of the centre of mass are given as follows:

$$v_y = a\dot{\theta} \tag{3.44}$$

$$\dot{v}_y = a\ddot{\theta} \tag{3.45}$$

By substituting Equation (3.45) into Equation (3.42), it yields:

$$a\ddot{\theta} = -v_x\dot{\theta} + \frac{(F_{yl} + F_{yr})}{m} \tag{3.46}$$

$$m(a\ddot{\theta} + v_x\dot{\theta}) = F_{yl} + F_{yr} \tag{3.47}$$

By substituting Equation (3.47) into Equation (3.43) and solve for $\ddot{\theta}$ as in the following equations:

$$\ddot{\theta} = -\frac{m(a\ddot{\theta} + v_x\dot{\theta})a}{I_c} + \frac{(F_{xr} - F_{xl})L}{I_c} \tag{3.48}$$

$$\ddot{\theta}I_c = -m(a\ddot{\theta} + v_x\dot{\theta})a + (F_{xr} - F_{xl})L \tag{3.49}$$

$$\ddot{\theta}I_c + ma^2\ddot{\theta} = (F_{xr} - F_{xl})L - v_x\dot{\theta}a \tag{3.50}$$

$$\ddot{\theta} = \frac{(F_{xr} - F_{xl})L}{ma^2 + I_c} - \frac{mav_x\dot{\theta}}{ma^2 + I_c} \tag{3.51}$$

Likewise, by substituting Equation (3.44) into Equation (3.41), the following equation is obtained:

$$\dot{v}_x = a\dot{\theta}^2 + \frac{(F_{xr} + F_{xl})}{m} \tag{3.52}$$

The torques of the left and right wheels can be determined as follows:

$$T_r = rF_{xr} \tag{3.53}$$

$$T_l = rF_{xl} \tag{3.54}$$

By substituting Equations (3.53) and (3.54) into Equations (3.51) and (3.52) and re-arranging the result, main dynamic equations of the differential unmanned ground vehicle considering

the non-holonomic constraints are obtained based on the Newtonian dynamic approach as follows:

$$m\dot{v} - ma\dot{\theta}^2 = \frac{(T_r + T_l)}{r} \tag{3.55}$$

$$(ma^2 + I_c)\ddot{\theta} + mav\dot{\theta} = \frac{(T_r - T_l)L}{r} \tag{3.56}$$

The above equations can be transformed to the following matrix form as follows:

$$\begin{bmatrix} m & 0 \\ 0 & ma^2 + I_c \end{bmatrix}\begin{bmatrix} \dot{v} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & -ma\dot{\theta} \\ ma\dot{\theta} & 0 \end{bmatrix}\begin{bmatrix} v \\ \dot{\theta} \end{bmatrix} = \frac{1}{r}\begin{bmatrix} 1 & 1 \\ L & -L \end{bmatrix}\begin{bmatrix} T_r \\ T_l \end{bmatrix} \tag{3.57}$$

The matrix elements are stated as follows:

$$\overline{M}(q) = \begin{bmatrix} m & 0 \\ 0 & ma^2 + I_c \end{bmatrix} \tag{3.58}$$

$$\overline{C}(q, \dot{q}) = \begin{bmatrix} 0 & -ma\dot{\theta} \\ ma\dot{\theta} & 0 \end{bmatrix} \tag{3.59}$$

$$B(q) = \frac{1}{r}\begin{bmatrix} 1 & 1 \\ L & -L \end{bmatrix} \tag{3.60}$$

The calculations of moment of inertia is attached in the Appendix and the relevant physical parameters of the unmanned ground vehicle are illustrated in Table 3.1.

Table 3.1 Parameters of the unmanned ground vehicle.

| **Parameter** | **Description** | **Value** | **Unit** |
|---|---|---|---|
| r | Wheel radius | 0.12 | m |
| L | Distance between the drive wheel and the axis of symmetry | 0.20 | m |
| a | Distance between the centre of mass and drive wheel axis | 0.10 | m |
| m | Mass of the vehicle with driving wheels and motors | 5 | kg |
| $I_c$ | Moment of inertia about the centre of mass | 0.0427 | $Kg.m^2$ |

### *3.4.3 Actuators Modelling*

An actuator is an electrical system such as a servo or a direct current (DC) motor that drives the mechanical part of a robotic system. Each system might have multiple number of actuators based on its mechanism. The actuators receive a control signal directly from a control system to be activated in order to drive wheels into a specified motion. Otherwise, they are in

idle modes. DC motors are used as actuators for driving the UGV in this work. The electrical circuit and mechanical part of the DC motor is depicted in Fig. 3.8.

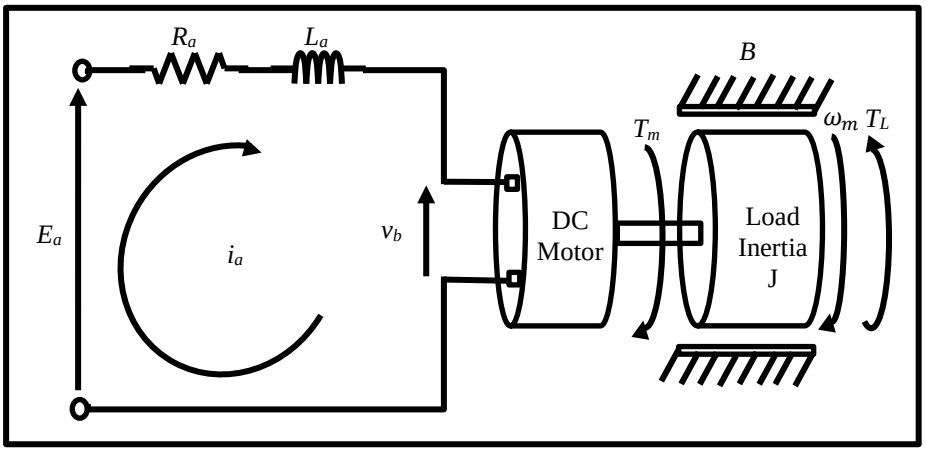



Fig. 3.8 Electrical circuit and mechanical part of a DC motor.

The basic model of a DC motor based on the electric circuit is derived and given in the following equations:

$$T_m = k_m i_a \tag{3.61}$$

$$v_b = k_b \omega_m \tag{3.62}$$

$$E_a = R_a i_a + L_a \frac{di_a}{dt} + v_b \tag{3.63}$$

$$T_m = J_a \frac{d\omega}{dt} + B\omega \tag{3.64}$$

where,

$\omega_m$ - Angular speed of motor,

$i_a$ - Armature current,

$E_a$ - Armature terminal voltage,

$v_b$ - Back electromotive force (e.m.f) voltage,

$J_a$ - Motor inertia,

$T_m$ - Motor torque.

By taking Laplace transform of the four equations above, the following equations are obtained:

$$T_m(s) = k_m . I_a(s) \tag{3.65}$$

$$V_b(s) = k_b \omega_m(s) \tag{3.66}$$

$$T_m(s) = J_a . s . \omega_m(s) + B . \omega_m(s) \tag{3.67}$$

$$E_a(s) = R_a I_a(s) + L_a . s . I_a(s) + V_b(s) \tag{3.68}$$

From Equation (3.67), the following equations can be found:

$$T_m(s) = [J_a.s + B]\omega_m(s) \tag{3.69}$$

$$\omega_m(s) = \left[\frac{1}{J_a.s + B}\right] T_m(s) \tag{3.70}$$

Similarly, Equation (3.68) can be re-arranged to find the following equations:

$$E_a(s) = (R_a + L_a.s).I_a(s) + V_b(s) \tag{3.71}$$

$$E_a(s) - V_b(s) = (R_a + L_a.s).I_a(s) \tag{3.72}$$

$$I_a(s) = \left[\frac{1}{R_a + L_a.s}\right][E_a(s) - V_b(s)] \tag{3.73}$$

From Equations (3.65), (3.66), (3.70) and (3.73), the main transfer function between the applied voltage and the angular velocity can be found as follows:

$$\frac{\omega_m(s)}{E_a(s)} = \frac{k_m}{J_a.L_a.s^2 + (J_a.R_a + BL_a).s + (B.R_a + k_b k_m)} \tag{3.74}$$

The physical parameters of the actuator are given in Table 3.2 below:

Table 3.2 Physical parameters of the actuators.

| **Parameter** | **Description** | **Value** | **Unit** |
|---|---|---|---|
| $V$ | Nominal voltage | 12 | V |
| $N_o$ | No-load speed | 200 | RPM |
| $N_r$ | Rated speed | 163 | RPM |
| $R_a$ | Resistance of the armature winding | 0.5 | Ω |
| $L_a$ | Inductance in the motor winding | 0.1 | H |
| $J_a$ | Moment of inertia | 0.00036 | Kg.m$^2$ |
| $k_m$ | Torque constant | 0.268 | N.m/A |
| $k_b$ | Back e.m.f. constant | 0.01 | V.s/rad |
| $B$ | Viscous friction | 0.001 | N.m.s |
| $T_m$ | Motor rated torque | 0.0764 | N.m |
| $i_o$ | No-load current | 0.115 | A |
| $i_a$ | Rated current | 0.285 | A |

## 3.5 Design PID Controller for Trajectory Tracking

In this section, the modelled system is conducted based on a so called proportional integral derivative (PID) controller. Simulation results will be accomplished to investigate how efficiently the system satisfies the operating condition and how far the performance from the required targets that reach a minimum tracking error. Nonetheless, different controllers will be proposed and integrated with the implemented model to explore a better response and minimising the error. The PID has been widely applied in many industrial applications(Chaudhary and Ohri, 2016; Pedro et al., 2016; Tian et al., 2014). Although PID controller has a simple structure, it has been approved that it capable of performing required functions efficiently. The PID ccontroller is designed to stabilise and enhance the performance of the system under certain conditions. However, in general, each controller is designed for a specific situation or scenario and is effective under these particular conditions. A PID controller improves the transient response of a system by reducing the overshoot and settling time of a system. The main reason to develop advanced methods to design PID controllers is the significant impact on the performance improvement. The performance index adopted for problem formulation is settling time, overshoot and oscillations. The primary design goal is to obtain a minimised tracking error by optimally selecting the PID controller parameters. The proportional integral differential equation governors the control action of a PID controller is given by:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{3.75}$$

The block diagram of the PID controller and the unmanned ground vehicle is depicted in Fig. 3.9 below.

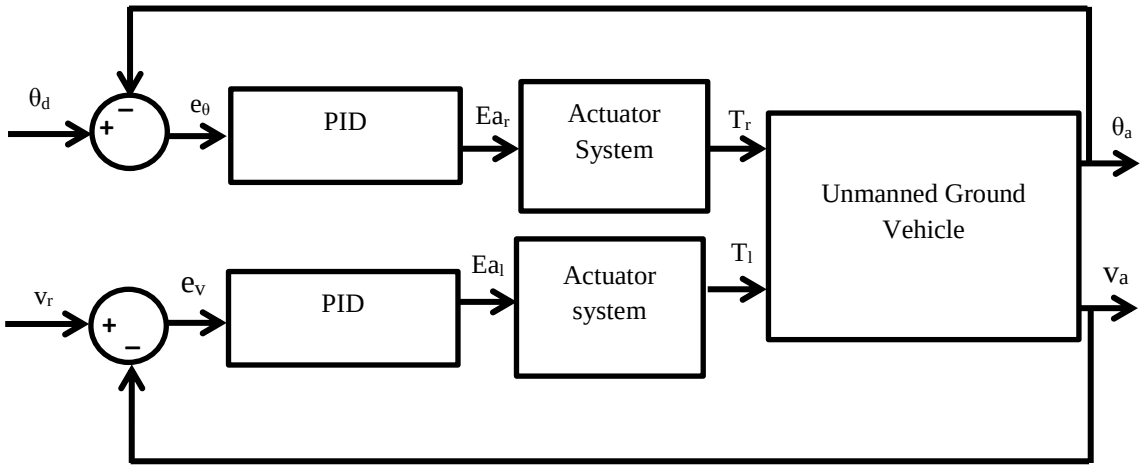



Fig. 3.9 Block diagram of PID controller and UGV.

The PID controllers are tuned based on self-tuning procedure using Ziegler-Nichols method (Bhatti et al., 2016), the optimum response is obtained based on reaching the parameters of PID controllers. The tuning procedure is achieved using control system toolbox based on MATLAB environment (Turevskiy, 2016).The parameters of the traditional PID controller for both orientation and velocity are given in Table 3.3 and Table 3.4, respectively. They represent the optimal parameters after tuning process of the system.

Table 3.3 Parameters of the traditional PID controller for UGV's orientation.

| Parameters | Proportional constant ($Kp_1$) | Integral constant ($Ki_1$) | Derivative constant ($Kd_1$) |
|---|---|---|---|
| Value | 1.754 | 1.830 | 12.904 |

Table 3.4 Parameters of the traditional PID controller for UGV's velocity.

| Parameters | Proportional constant ($Kp_2$) | Integral constant ($Ki_2$) | Derivative constant ($Kd_2$) |
|---|---|---|---|
| Value | 5.378 | 7.027 | 0.293 |

## 3.6 Simulation Results

In this section, four desired trajectories are considered as inputs for the implemented model and control system to investigate the trajectory tracking error. It demonstrates the behaviour of the system based on the traditional PID controller in terms of vehicle orientation, velocity control and tracking error. The three comparisons are conducted to demonstrate the performance and the effectiveness of the two PID controllers. In additional, this investigates the capability of tracking any trajectories with continuous and non-continuous gradients. These comparisons are made under identical conditions in order to specify the differences and similarities of using different trajectories. Consequently, the advantages and disadvantages of the utilised control methodology can be determined.

### *3.6.1 Linear Trajectory*

A linear trajectory can be simply generated based on providing a constant value for the desired orientation for a specific given time. For instance, an orientation of $\theta_d = \pi/4\ [rad]$ is used for interval $0 \leq t \leq 40\ [sec]$. As shown in Fig. 3.10, it can be clearly noticed that the

desired generated trajectory is a linear route that is a reference input to the system. However, the actual obtained output has a bumpy path alongside the desired linear trajectory. Hence, this confirms that the modelled system behaves in a proper manner to track the pre-defined input. However, it encounters a difficulty to adapt itself to guide the UGV precisely to track the given trajectory. Similarly, the relationship between the desired and actual orientation is shown in Fig. 3.11. Fig. 3.12 demonstrates that the orientation error is significantly high. Hence, this needs to be minimised drastically to reach an optimal trajectory tracking.

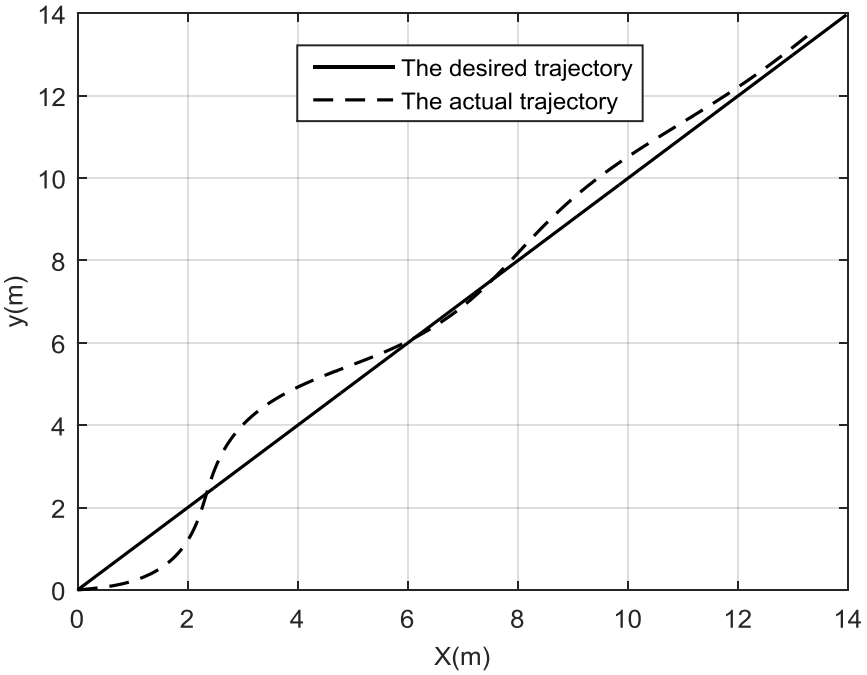



Fig. 3.10 Linear trajectories using PID controller.

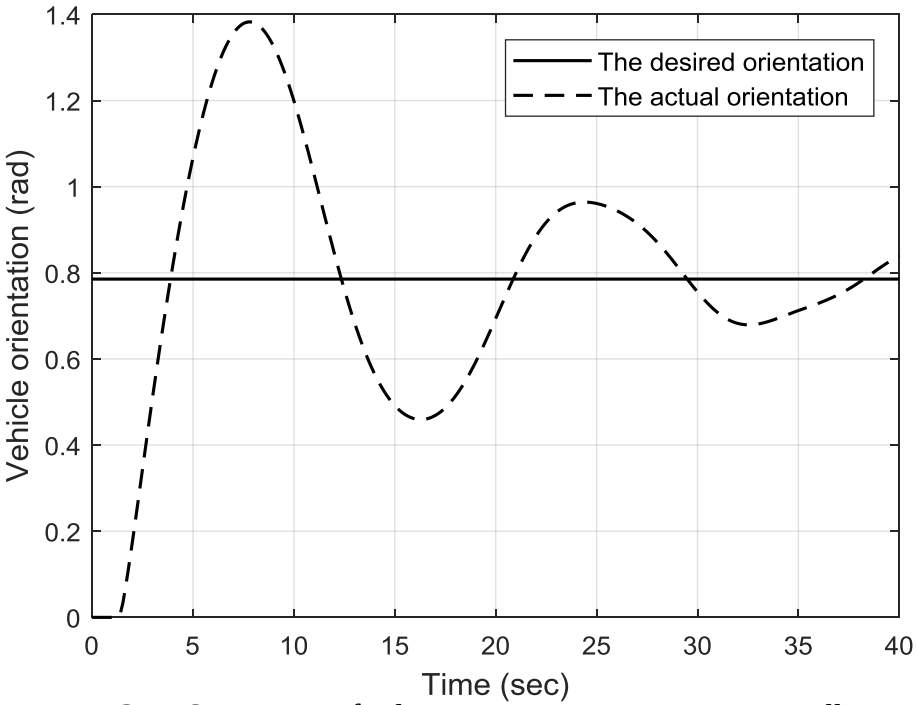



Fig. 3.11 Orientations for linear trajectory using PID controller.

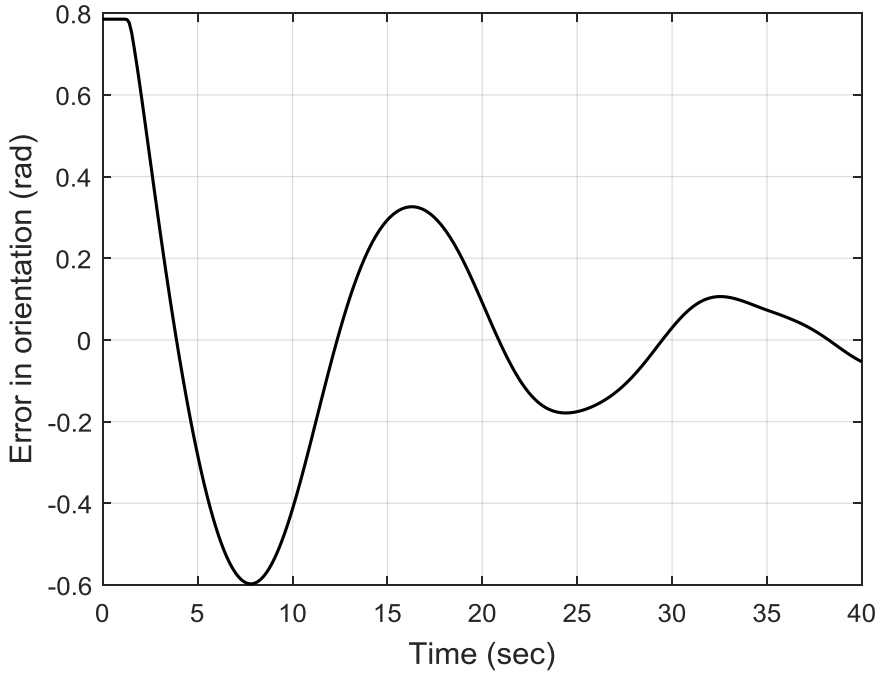



Fig. 3.12 Orientation error for linear trajectory using PID controller.

The above figures show that the system states reach their reference trajectories in about 40 seconds, which shows a relatively fast tracking response for the PID controller. The control actions needed from the UGV's wheel motors to provide such a tracking response are shown

in Fig. 3.13. This figure demonstrates that the system actuators can provide control efforts needed for such a trajectory tracking response. Therefore, the designed controller is applicable to the real platform.

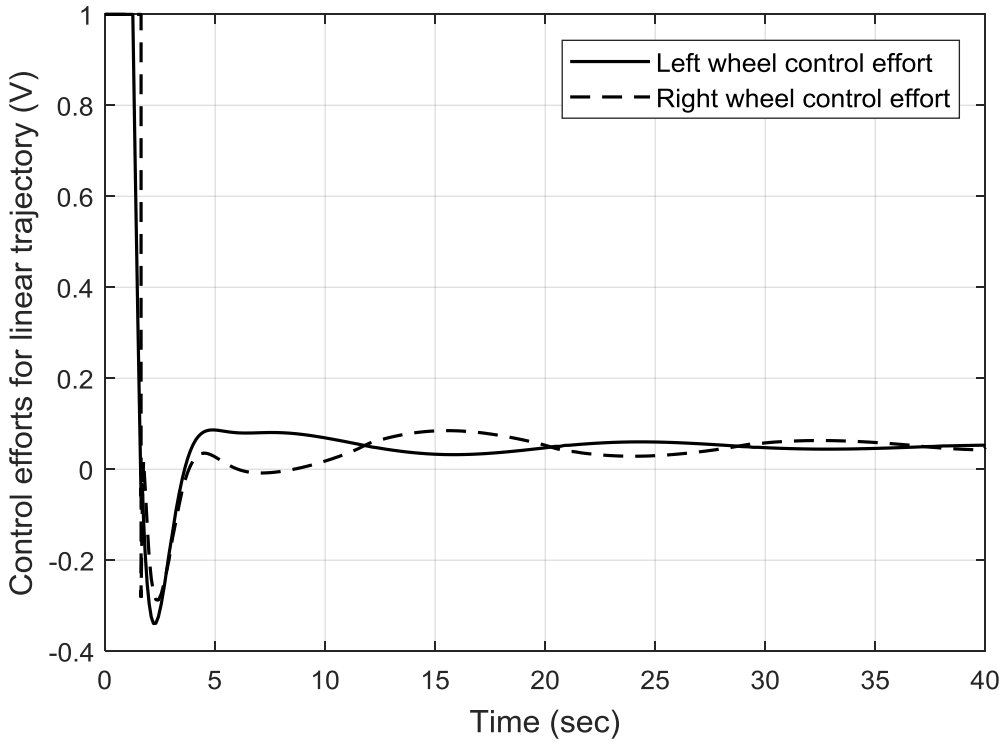



Fig. 3.13 Control efforts for linear trajectory using PID controller.

Another comparison is made based on the second PID controller for the velocity of the vehicle. The step input value is set at '0.5 m/s' as shown in Fig. 3.14. It has been noticed that the actual velocity is successfully reached the desired velocity. However, the overshoot and the settling time are still high and this demands designing a better control system to enhance the operational performance of the system. The error rate of the velocity is given in Fig. 3.15. The peak value shown in the first two seconds is normal because the velocity operation commences from standstill. Hence, the velocity is zero and this in turn leads to reach the maximum value of step input. Therefore, the influence of the error rate is considered when the velocity reaches the desired output at the steady state level.

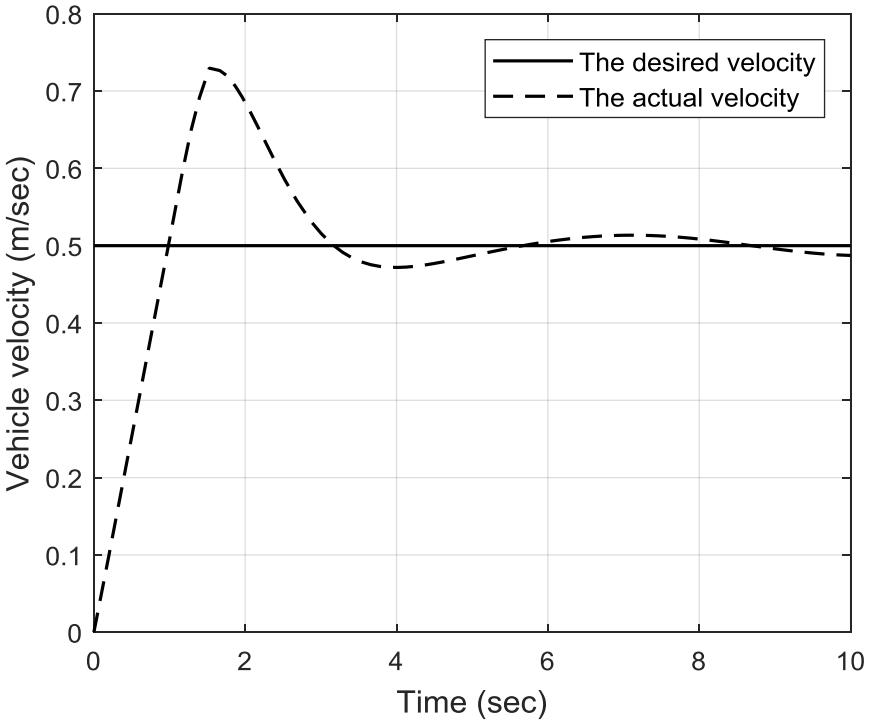



Fig. 3.14 Velocities for linear trajectory using PID controller.

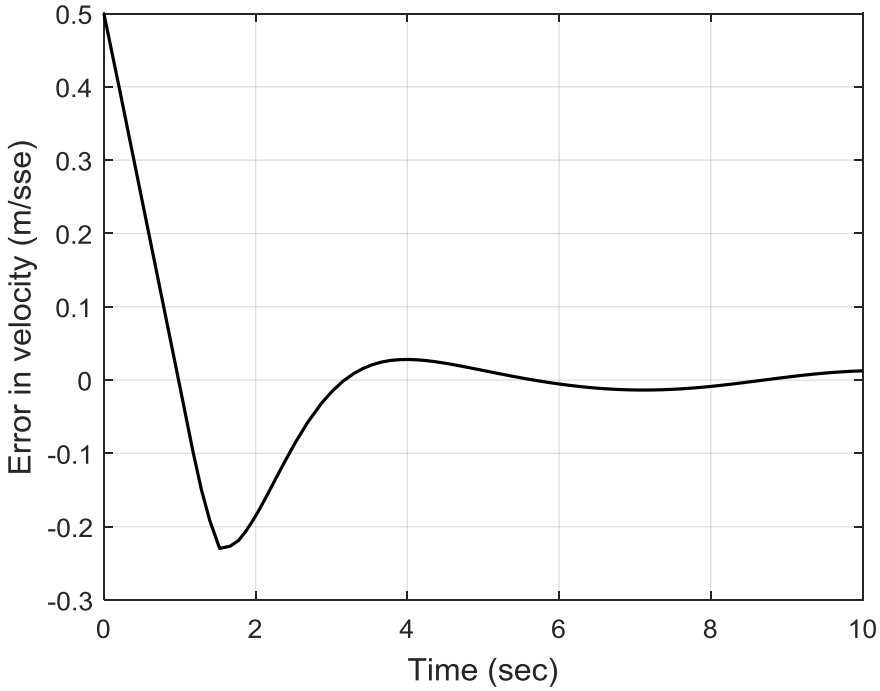



Fig. 3.15 Velocity error for linear trajectory using PID controller.

The last comparison is conducted based on the coordinates of both X and Y-axes. Figs. 3.16 and 3.17 demonstrate the desired coordinates of both X and Y-axes, respectively. The errors for both axes are depicted in Fig. 3.18 and again the tracking error is relatively high and this will be improved in the next chapter based on proposing a different control methodology.

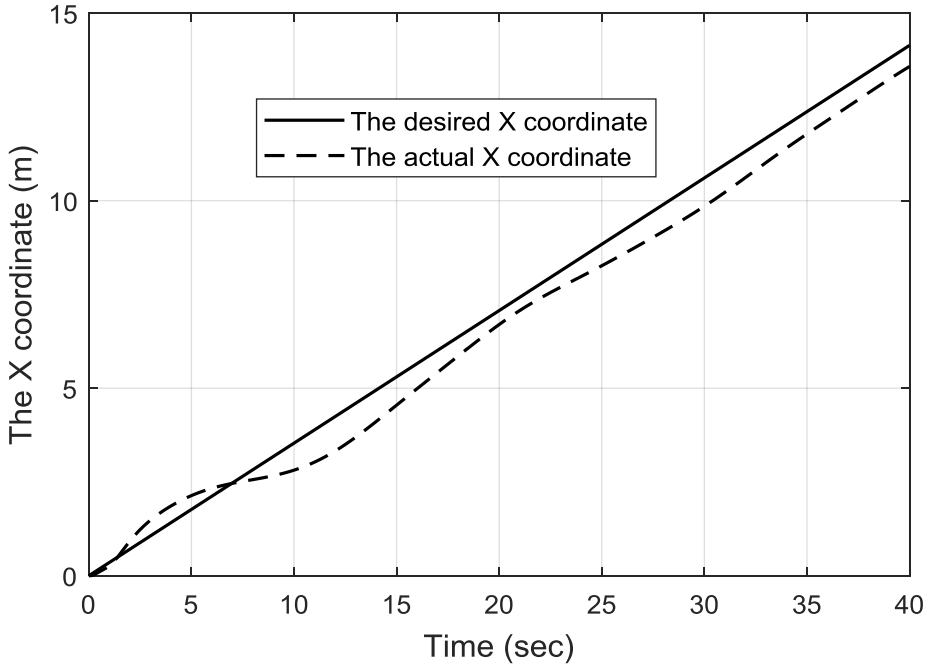



Fig. 3.16 X-coordinates for linear trajectory using PID controller.

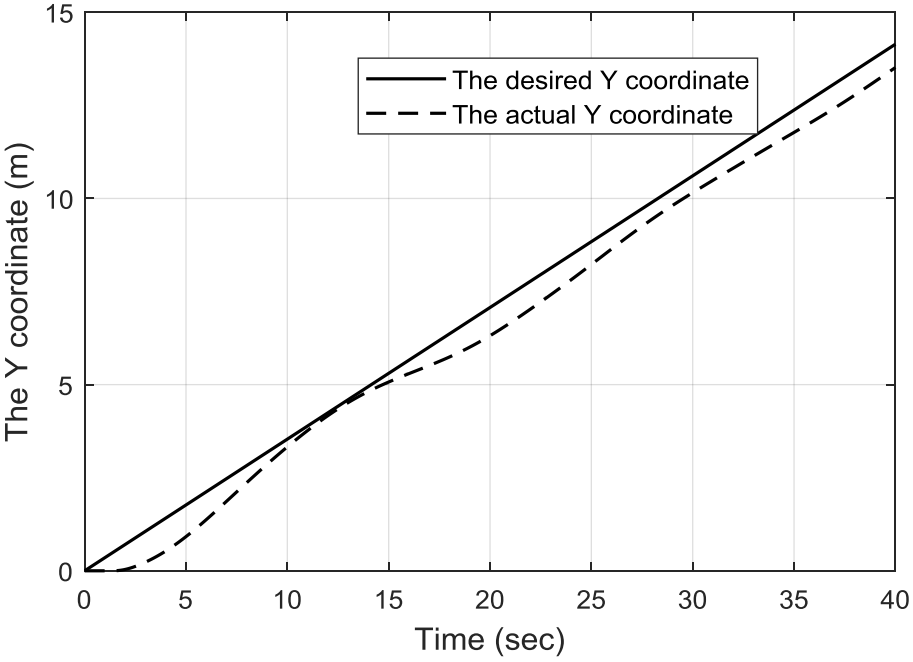



Fig. 3.17 Y-coordinates for linear trajectory using PID controller.

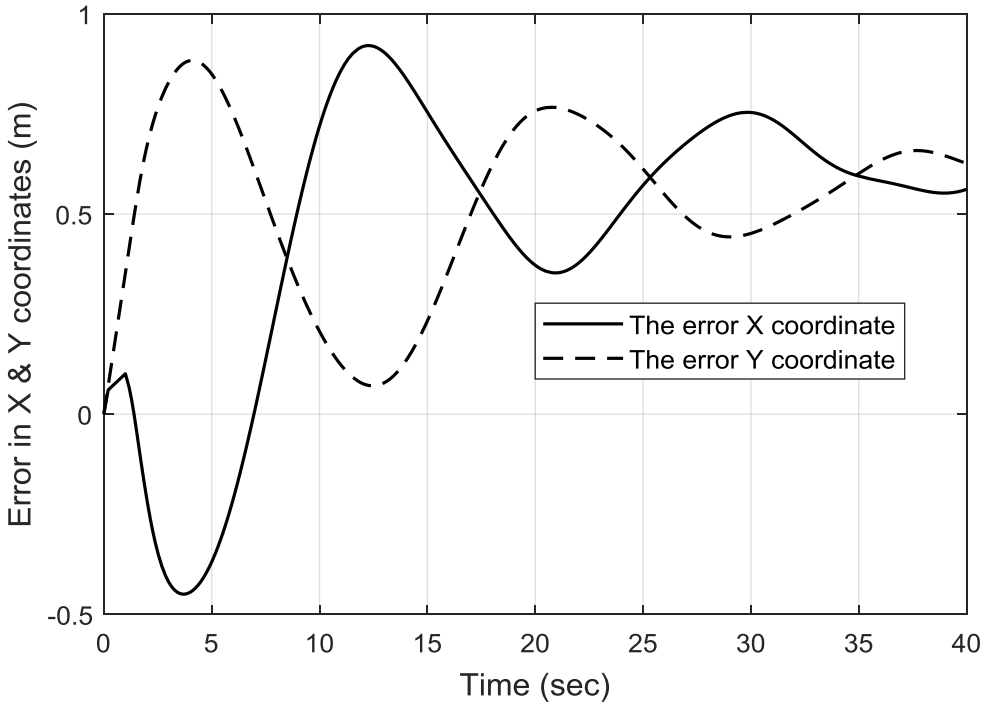



Fig. 3.18 Error in X and Y coordinates for linear trajectory using PID controller.

### *3.6.2 Circular Trajectory*

In this case, a circular trajectory has been generated. It demonstrates a different orientation that the UGV might confront comparing to a linear trajectory. The implementation process of this trajectory has been achieved based on the following equations. The input profile of the velocity is the same as in the linear trajectory. However, the actual velocity is expected to be slightly different due to the response of the new patterns of the circular trajectory. The simulation results for the desired, actual are shown in Fig. 3.19. The results of the desired and actual orientations are shown in Fig. 3.20. The error of the vehicle's orientation demonstrates a slight increment when the UGV commences the movement from the starting point. Later on, the error decays to the minimum as shown in Fig. 3.21. The orientation of the UGV is governed by the following equation and the given interval time by:

$$\theta_d = (2\pi t/-40)\ [rad\,],\ \ 0 \leq t \leq 40\ [sec\,] \tag{3.76}$$

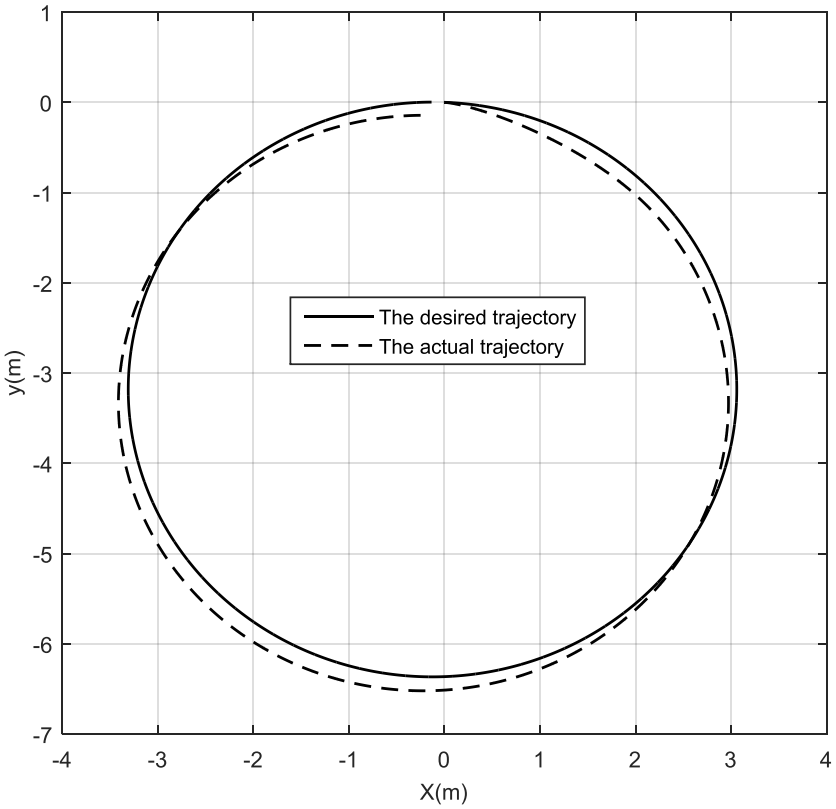



Fig. 3.19 Circular trajectories using PID controller.

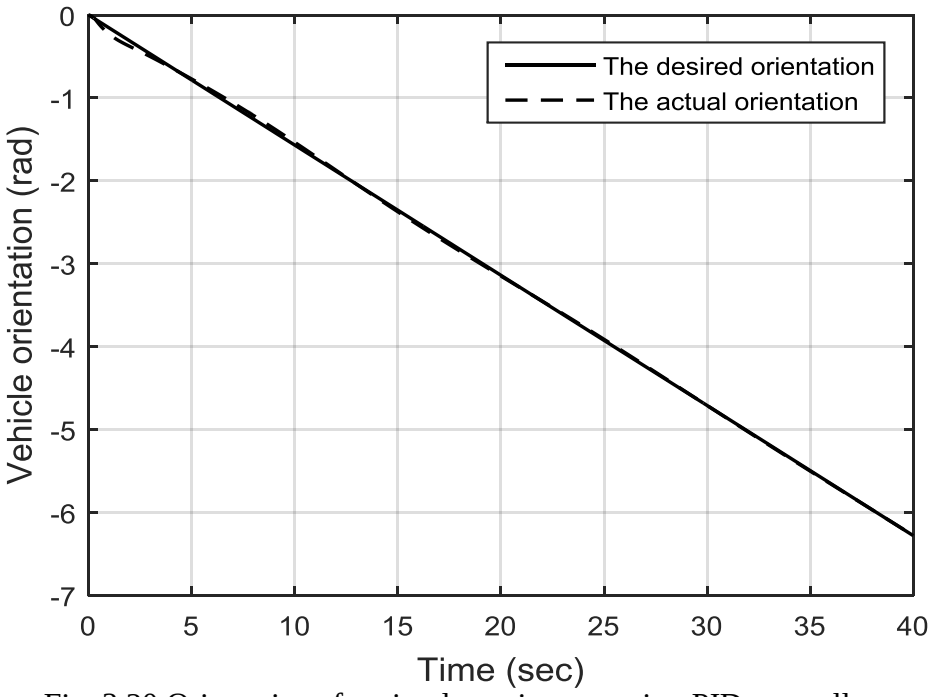



Fig. 3.20 Orientations for circular trajectory using PID controller.

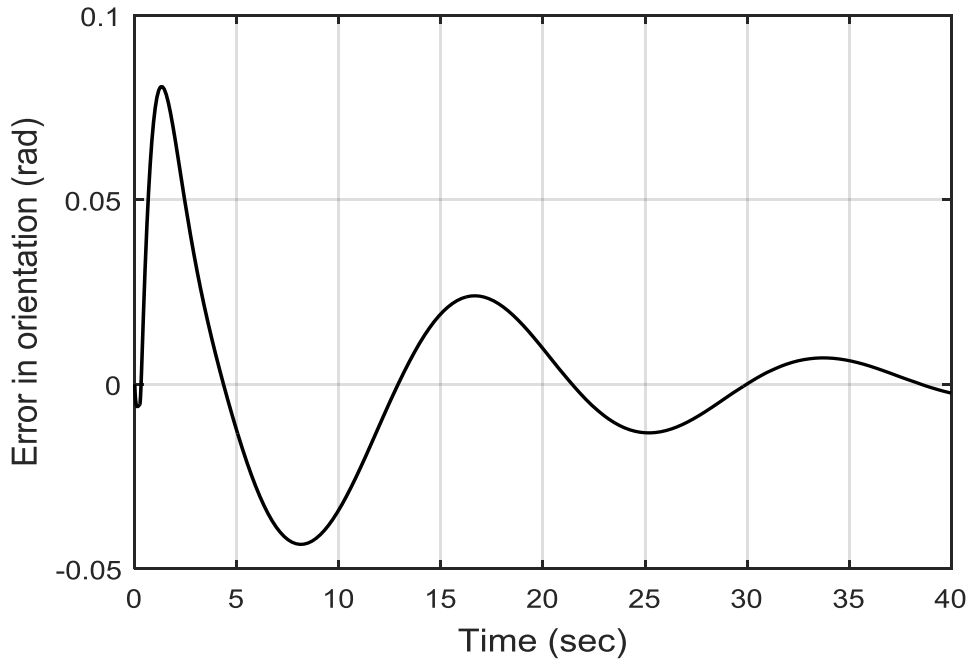



Fig. 3.21 Orientation error for circular trajectory using PID controller.

The control actions needed from the UGV's wheel motors to provide such a tracking response are shown in Fig. 3.22. It shows smooth response of system actuators, which can provide control efforts needed for such a trajectory tracking response. The desired and the actual velocities are obtained for such a circular trajectory as discussed previously. The actual velocity has an overshoot over the first five seconds and it has shortly decayed and reach the optimal value. This approves that the traditional PID controller has satisfied the input requirement. Although there is an overshoot and it cannot be avoided, this will be studied further based on a different control strategy to improve the response. The desired and actual velocity using the traditional PID controller is shown in Fig. 3.23 and the consequent error is depicted in Fig. 3.24. The trajectory tracking for X and Y coordinates are demonstrated in Fig. 3.25 and Fig. 3.26, respectively. The tracking error for both axes is illustrated in Fig. 3.27. It clearly shows that the tracking error has a maximum overshoot at the first five seconds. However, this is improved after a few seconds and the tracking error is reached the optimal value after 20 seconds.

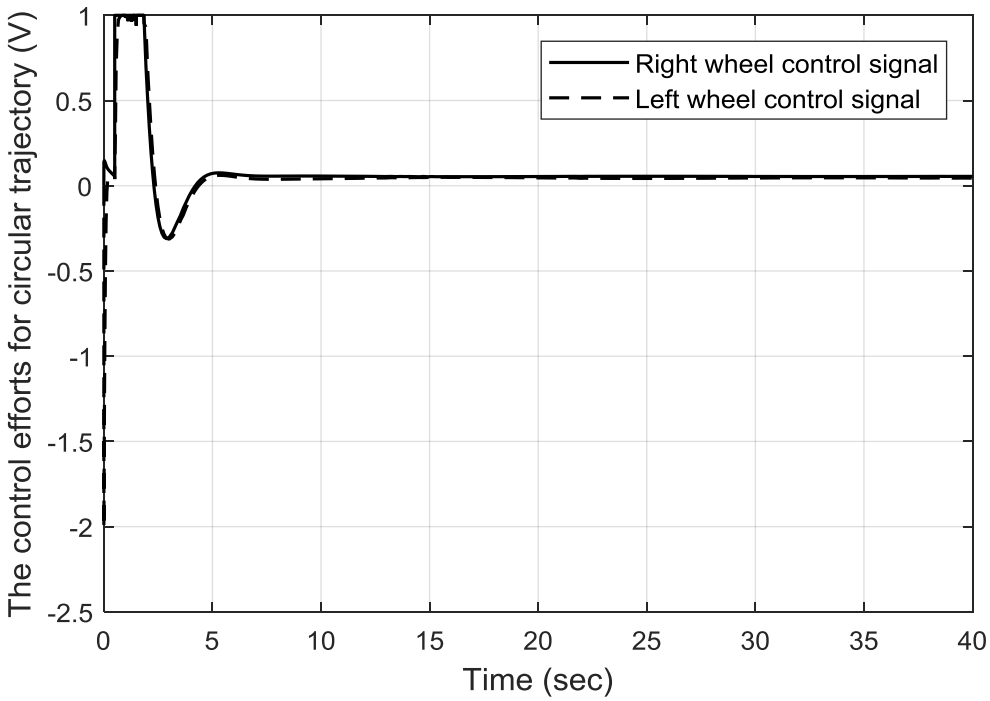



Fig. 3.22 Control efforts for circular trajectory using PID controller.

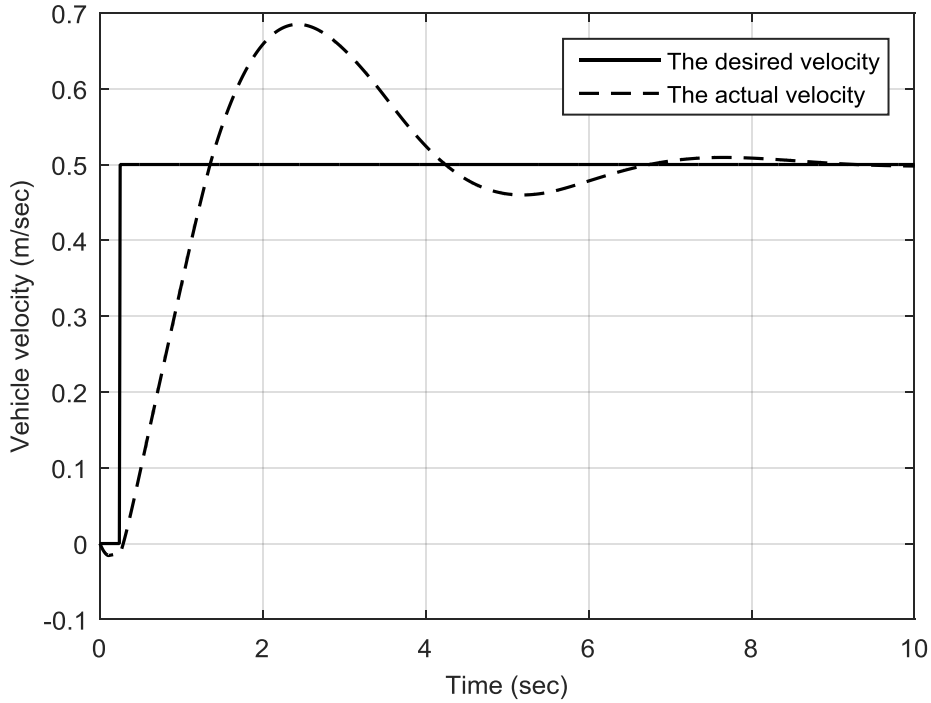



Fig. 3.23 Velocities for circular trajectory using PID controller.

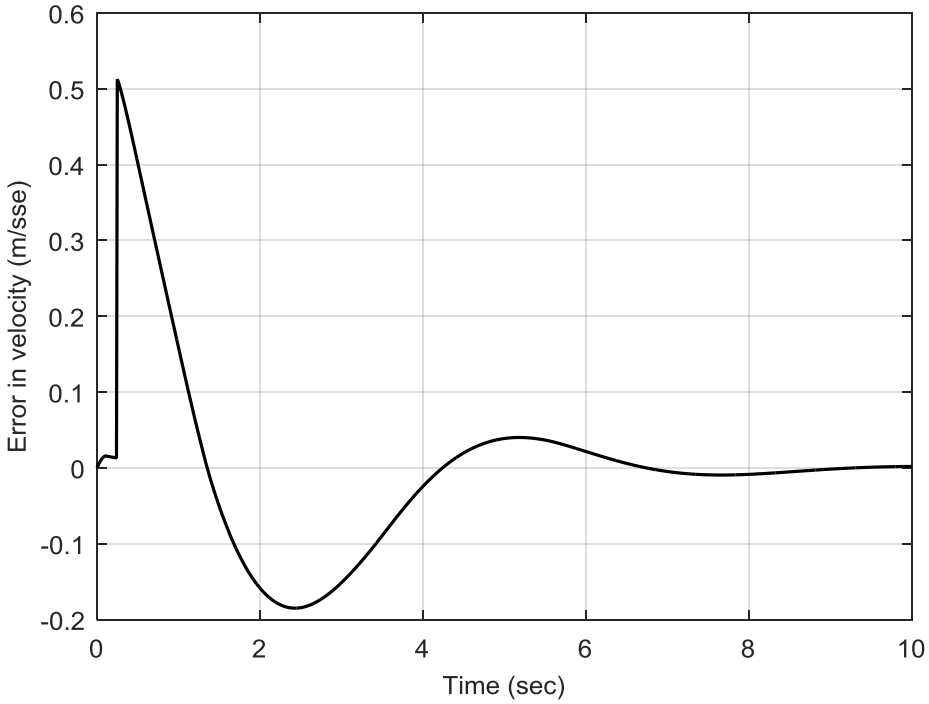



Fig. 3.24 Error in velocity for circular trajectory using PID controller.

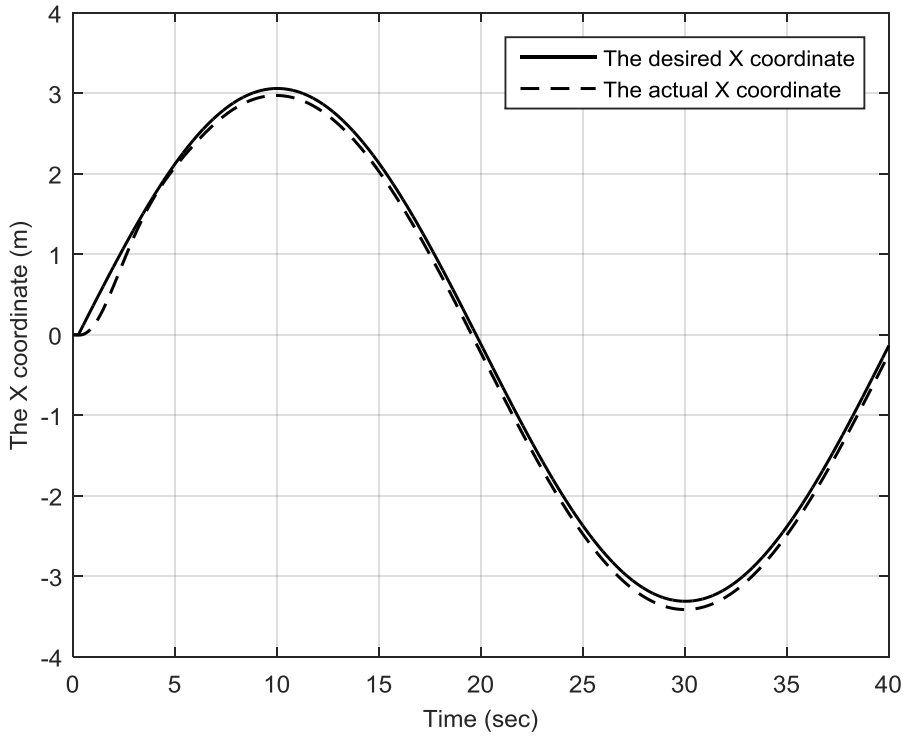



Fig. 3.25 X- coordinates for circular trajectory using PID controller.

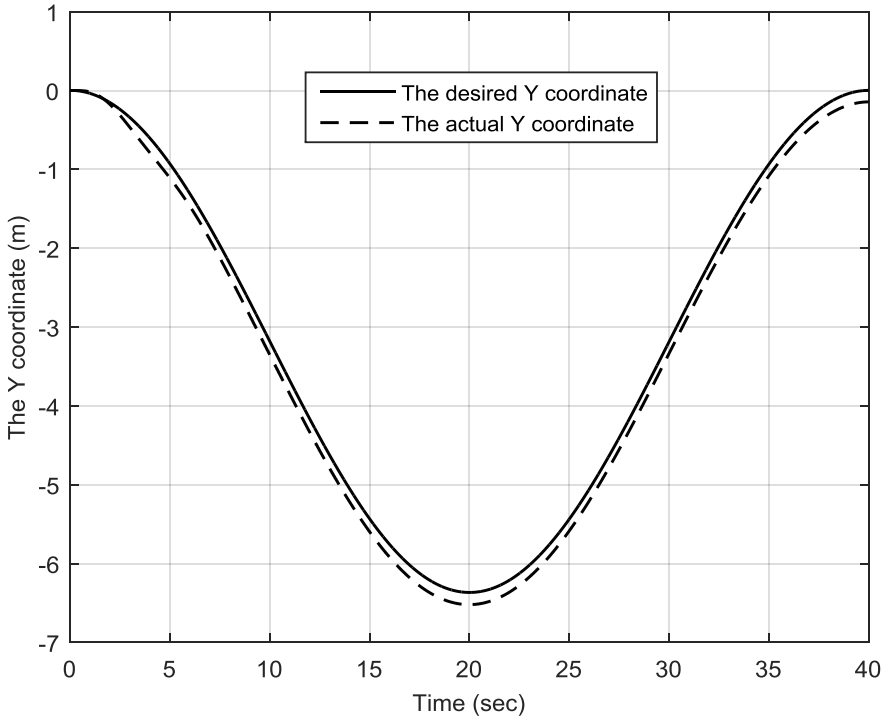



Fig. 3.26 Y- coordinates for circular trajectory using PID controller.

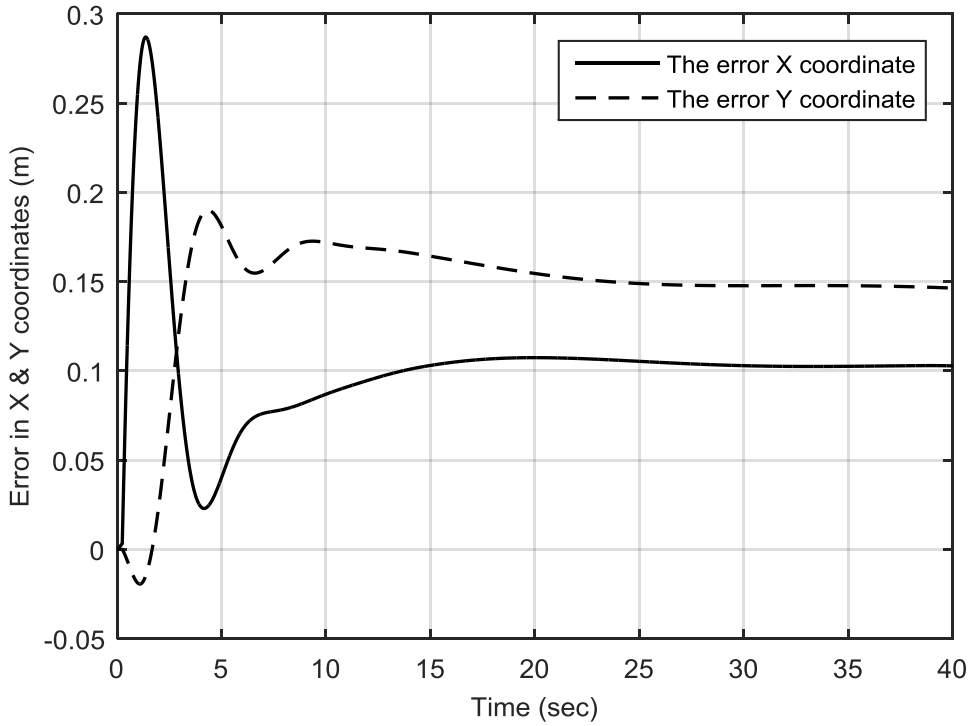



Fig. 3.27 Error in X and Y coordinates for circular trajectory using PID controller.

### *3.6.3 Leminscate Trajectory*

The lemniscate trajectory is another type of continuous gradient route. Although lemniscate trajectory's orientation is nearly similar to circular trajectory, it is a plane curve with a characteristic shape that it consists of two loops that meet at a central point. It demonstrates a roundabout motion because of changing the directions of coordinates whilst moving. The lemniscate trajectory is generated using the following equations:

$$x = \cos(\theta(t)) \tag{3.77}$$

$$y = \sin(2\theta(t)/2) \tag{3.78}$$

The time interval is $0 \leq t \leq 6.5\ [sec]$

The desired and actual lemniscate trajectories of the UGV are depicted in Fig. 3.28. It demonstrates a quite reasonable performance of trajectory tracking based on the used PID controller. The steering of the UGV based on the desired and actual orientation is illustrated in Fig. 3.29. The error between the desired and the actual orientation is demonstrated in Fig. 3.30.

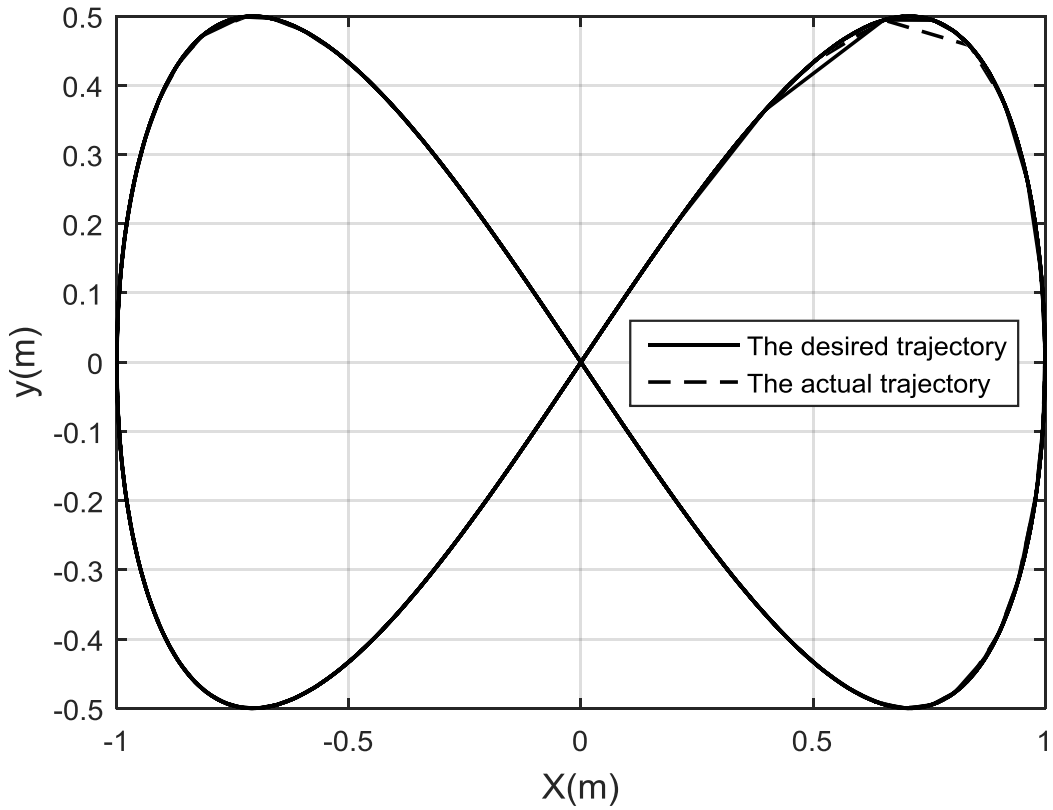



Fig. 3.28 Lemniscate trajectories using PID controller.

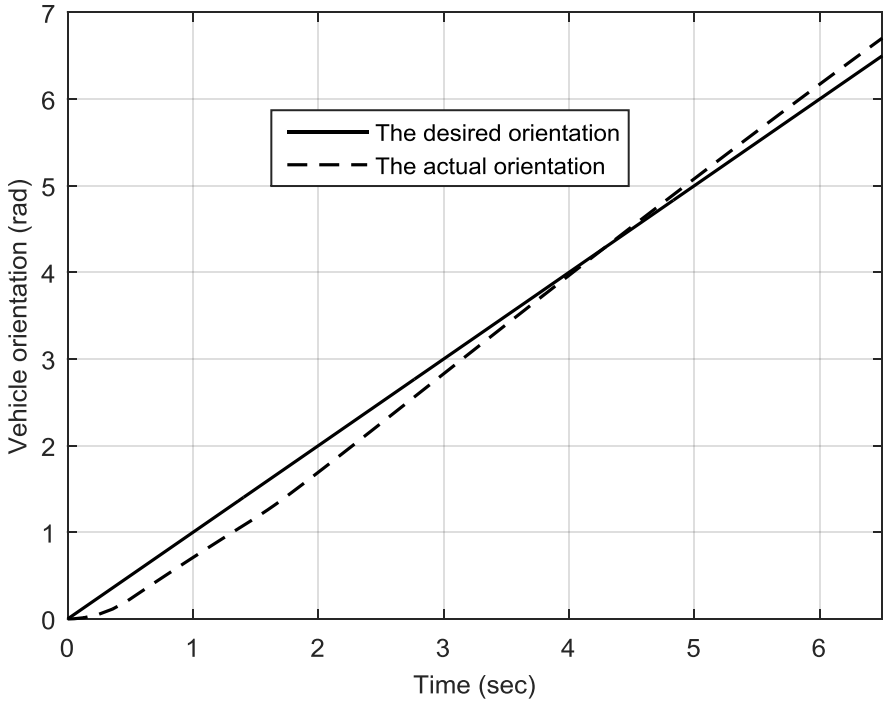



Fig. 3.29 Orientations for lemniscate trajectory using PID controller.

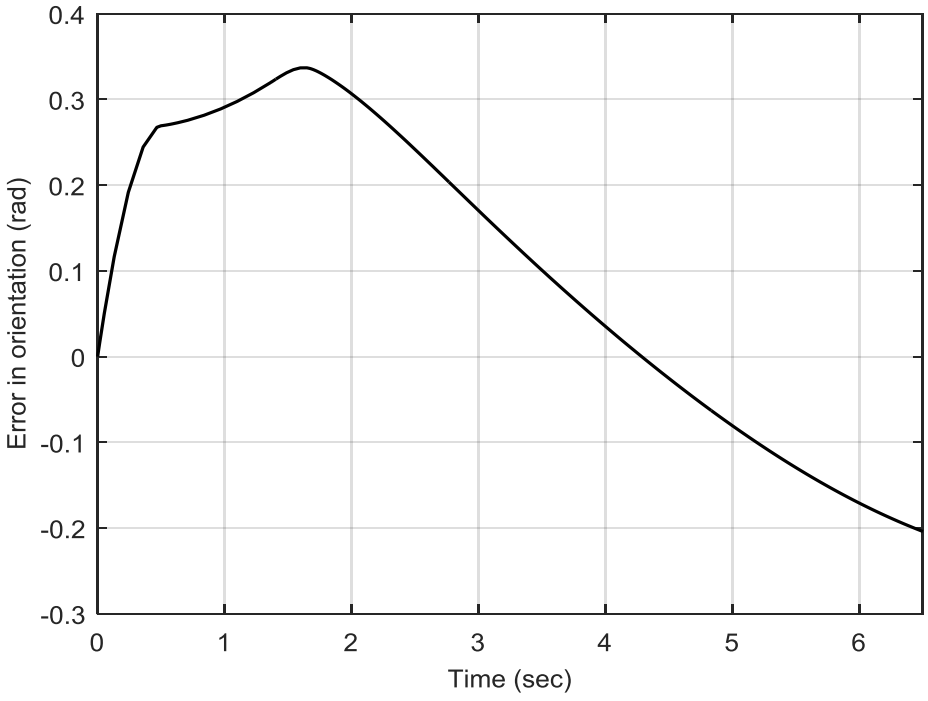



Fig. 3.30 Error in orientation for lemniscate trajectory using PID controller.

The control actions for leminsacte trajectory are given in Fig 3.31, which shows the efforts needed from the UGV's wheel motors to provide such a tracking response. The figure shows reasonable response of system actuators for such a trajectory.

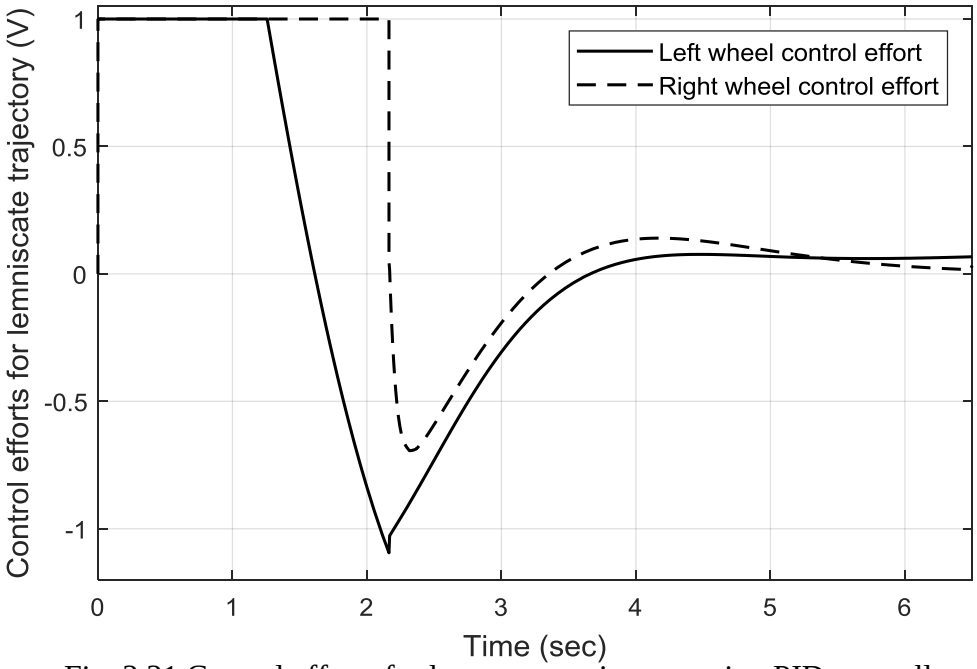



Fig. 3.31 Control efforts for lemnscate trajectory using PID controller.

In addition, the simulation results of the desired and actual velocity depicted in Fig. 3.32, have demonstrated the ability of the PID controller to control the velocity of the UGV. The desired velocity is a unit step function has a maximum amplitude of 0.5m/s. Although the actual velocity eventually reached the reference velocity, this shows an inappropriate velocity tracking due to remaining of velocity error whilst the movement of the UGV as shown in Fig. 3.33.

Moreover, the convergences between the desired and actual coordinates for both x and y-axes are illustrated in Fig. 3.34 and Fig. 3.35, respectively. The error of those convergences is shown in Fig. 3.36. The maximum tracking error in the X-coordinate trajectory is equal to 0.22m whilst the minimum tracking error is equal to -0.33m. Similarly, the maximum and the minimum tracking error in the Y-coordinate are equal to 0.6m and -0.34m. It is obvious that the tracking error for both coordinates is constantly diverged from reaching a minimum tracking value.

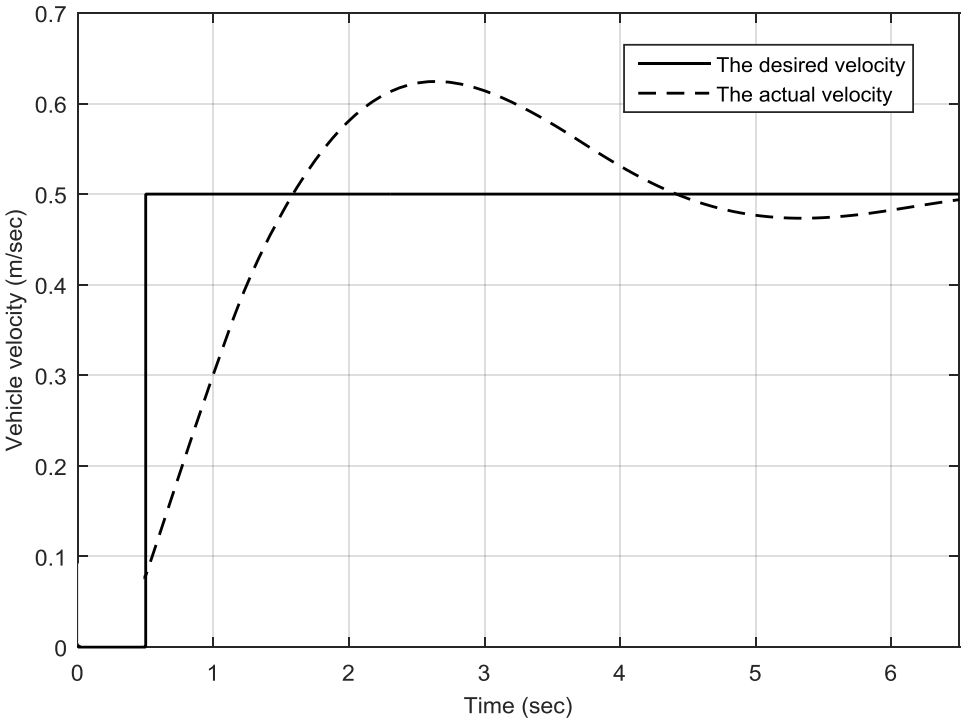



Fig. 3.32 Velocities for lemniscate trajectory using PID controller.

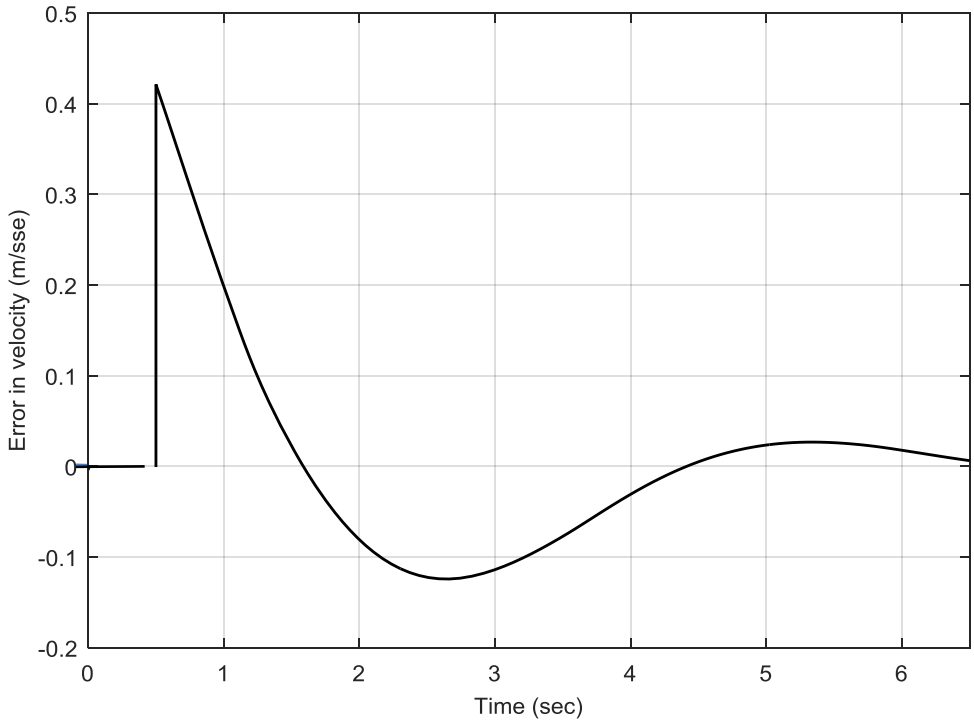



Fig. 3.33 Error in velocity for lemniscate trajectory using PID controller.

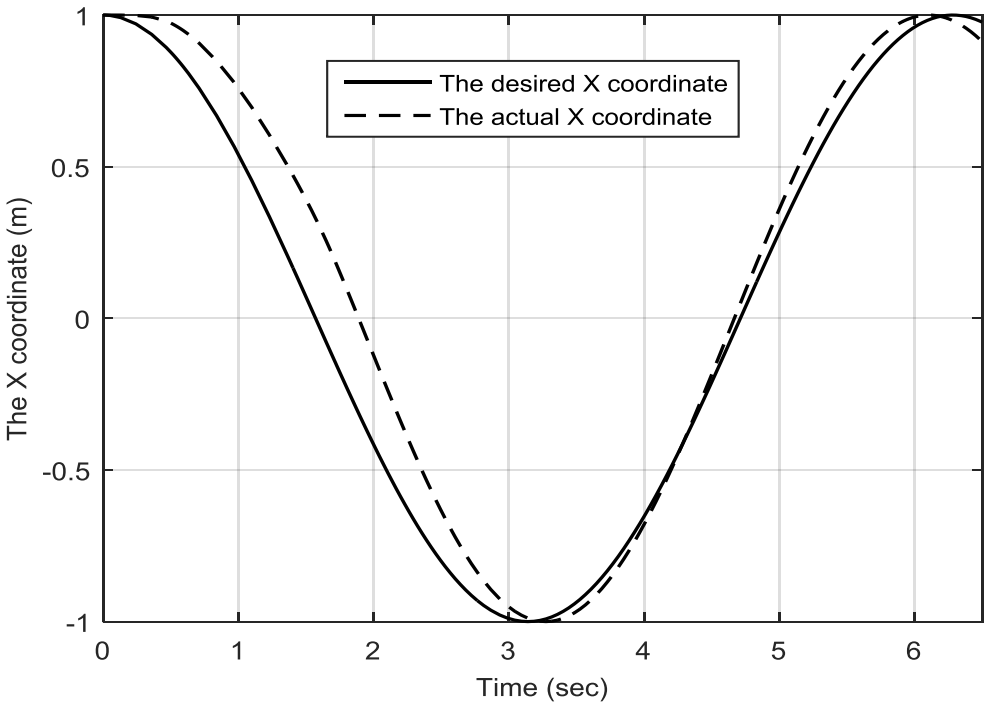



Fig. 3.34 X- coordinates for lemniscate trajectory using PID controller.

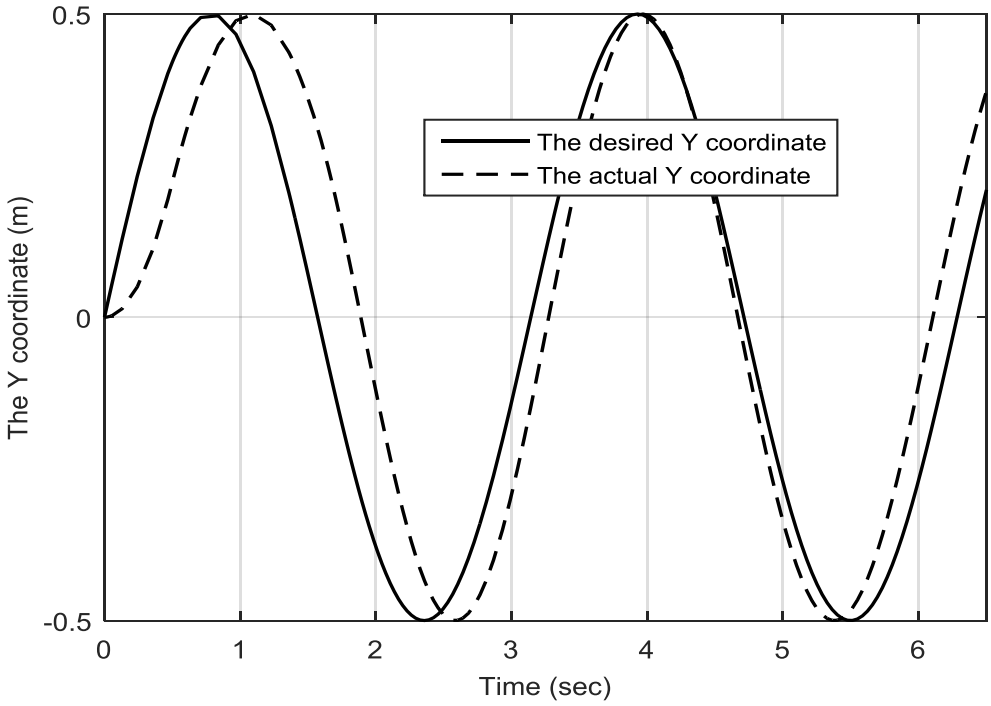



Fig. 3.35 Y- coordinates for lemniscate trajectory using PID controller.

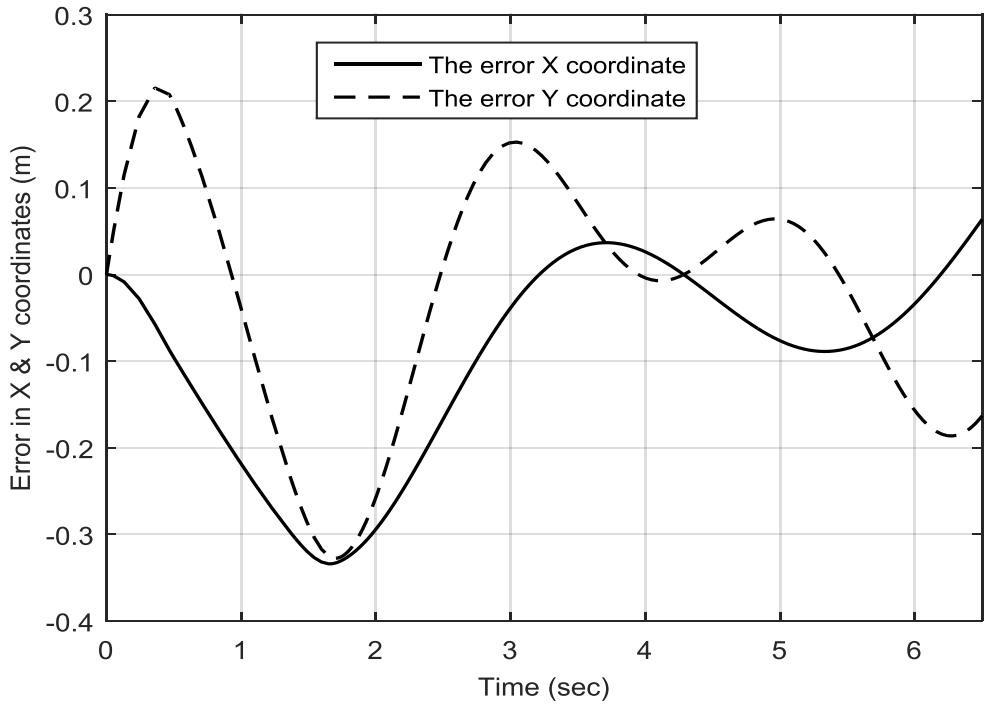



Fig. 3.36 Error in X and Y coordinates for lemniscate trajectory using PID controller.

### *3.6.4 Square Trajectory*

In this case study, a non-continuous gradient trajectory is generated, i.e. a square trajectory, using Equations 3.37 and 3.38. The equations are simply implemented as shown in Fig. 3.37 to generate an orientation of a square trajectory. This has been carried out to investigate the capability and the performance of the classic PID controller of obtaining a minimum trajectory tracking based on a non-continuous gradient. Fig. 3.38 demonstrates a generating of an accurate desired square trajectory. However, on the same figure, it is evident that the actual taken trajectory is diverged significantly from the desired trajectory. The reason for that belongs to the sudden changes of the orientation at the square corners. In addition, the PID controller has demonstrated a limited capability of adaptation for such circumstances. Hence, a new control strategy is needed to be designed for overcoming the large values of errors at each corner in particular. Fig. 3.39 illustrates the relationship between the desired and the actual orientation of the UGV. The error rate is remarkably high as shown in Fig. 3.40 comparing to the previous continuous gradient trajectories. The maximum value of the error is notably large and it equals to 1.6 m.

$$\theta_d = Theta1 + Theta2 + Theta3 + Theta4 \tag{3.79}$$

$$\left.\begin{array}{ll} Theta1 = \dfrac{\pi}{2} & 5 \le t \le 15 \\ Theta2 = \dfrac{\pi}{2} & 15 \le t \le 25 \\ Theta3 = \dfrac{\pi}{2} & 25 \le t \le 35 \\ Theta4 = \dfrac{\pi}{2} & 35 \le t \le 40 \end{array}\right\} \tag{3.80}$$

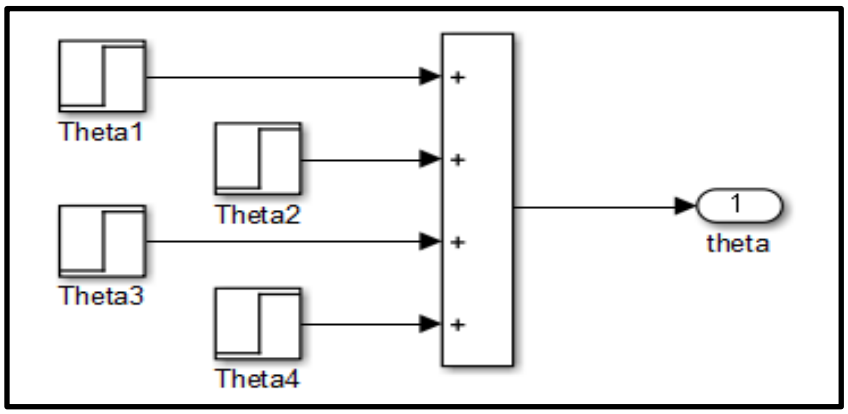



Fig. 3.37 Desired orientation for square trajectory.

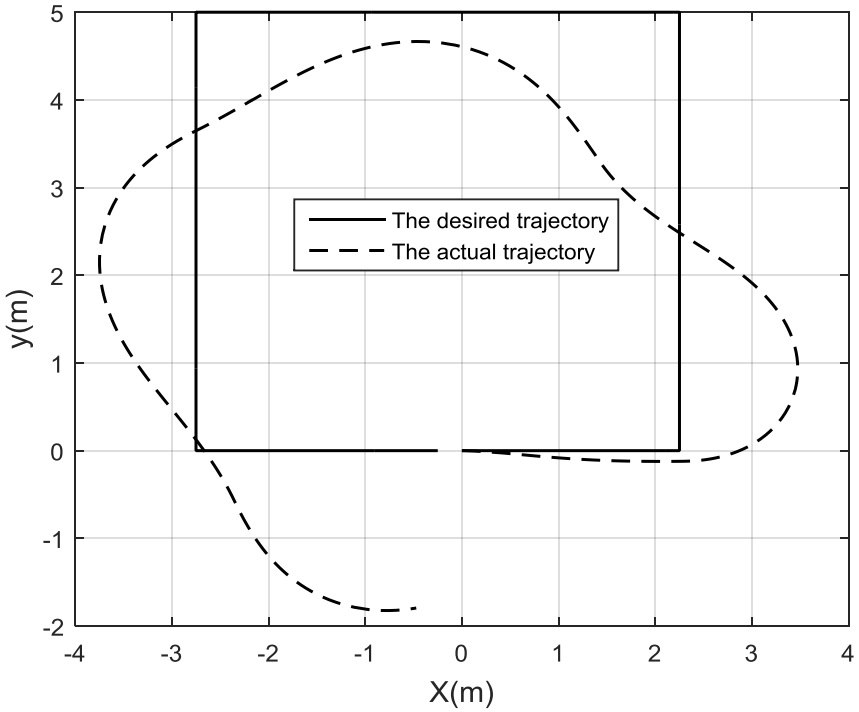



Fig. 3.38 Square trajectories using PID controller.

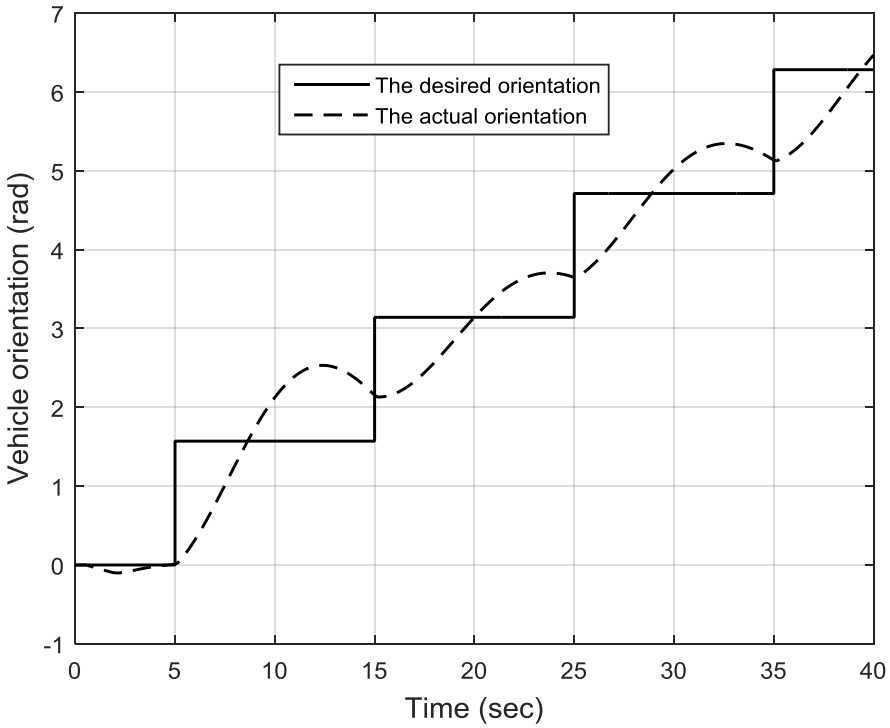



Fig. 3.39 Orientations for square trajectory using PID controller.

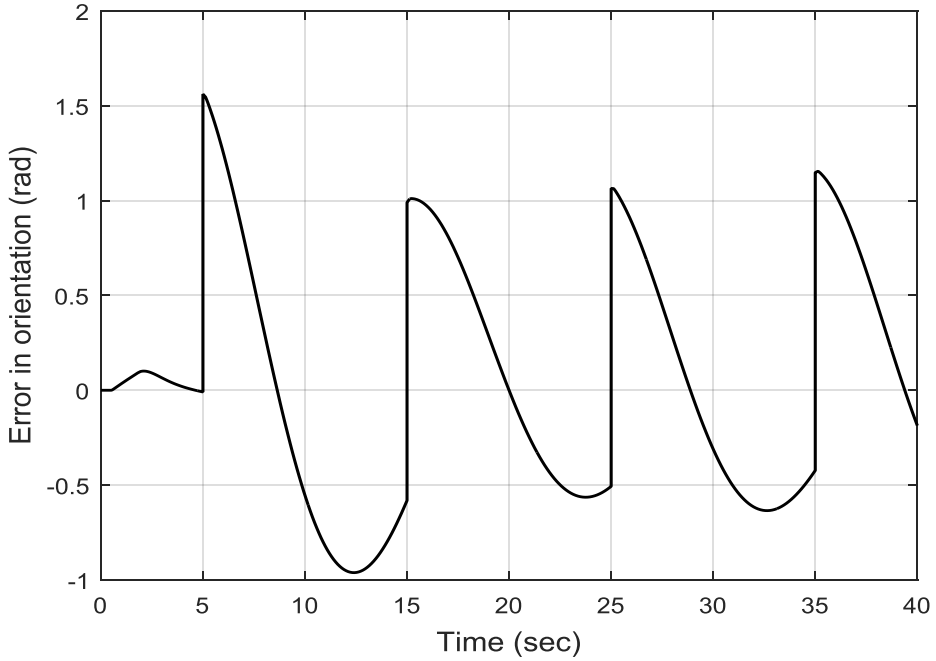



Fig. 3.40 Error in orientation for square trajectory using PID controller.

The control actions for square trajectory are relatively high when compared with the continuous gradient trajectories. This is because of a non-continuous gradient trajectory has sharp changes in its path and this require higher control efforts to overcome those changes. Fig. 3.41 demonstrates control effort of the UGV's wheel motors to provide such a tracking response.

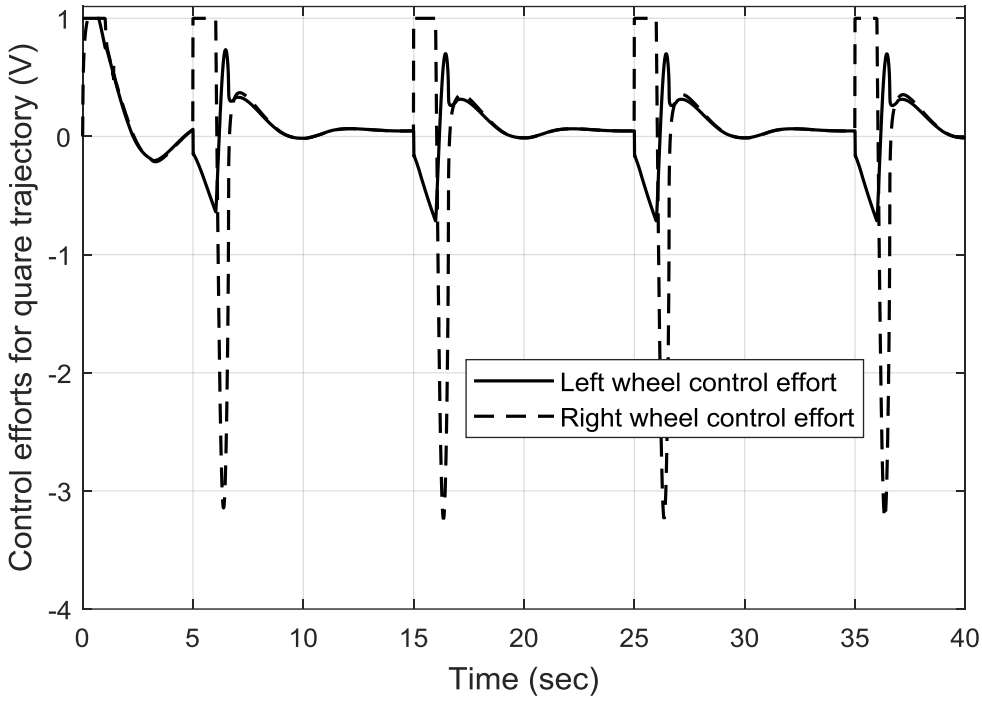



Fig. 3.41 Control efforts for square trajectory using PID controller.

The velocity profile of the square trajectory of the actual velocity demonstrates continuous changes due to the sudden changes of the motion whilst moving in the non-continuous gradient. However, those changes are still smooth values and without sharp spikes, as shown in Fig. 3.42. The error response is depicted in Fig. 3.43, which has an average maximum value of 0.17 m/s and an average minimum of -0.12 m/s.

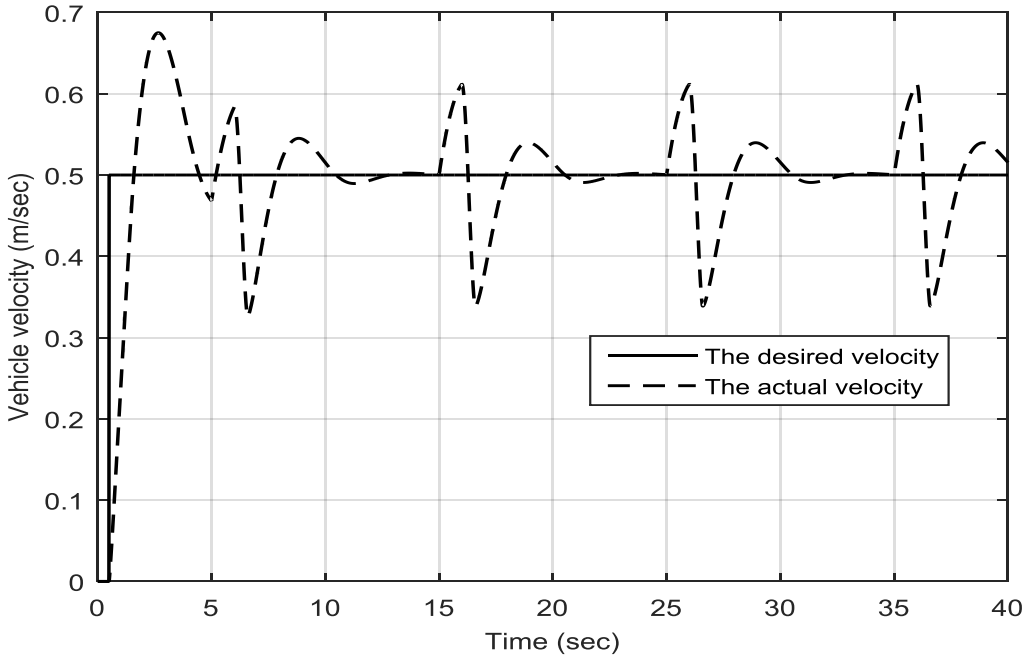



Fig. 3.42 Velocities for square trajectory using PID controller.

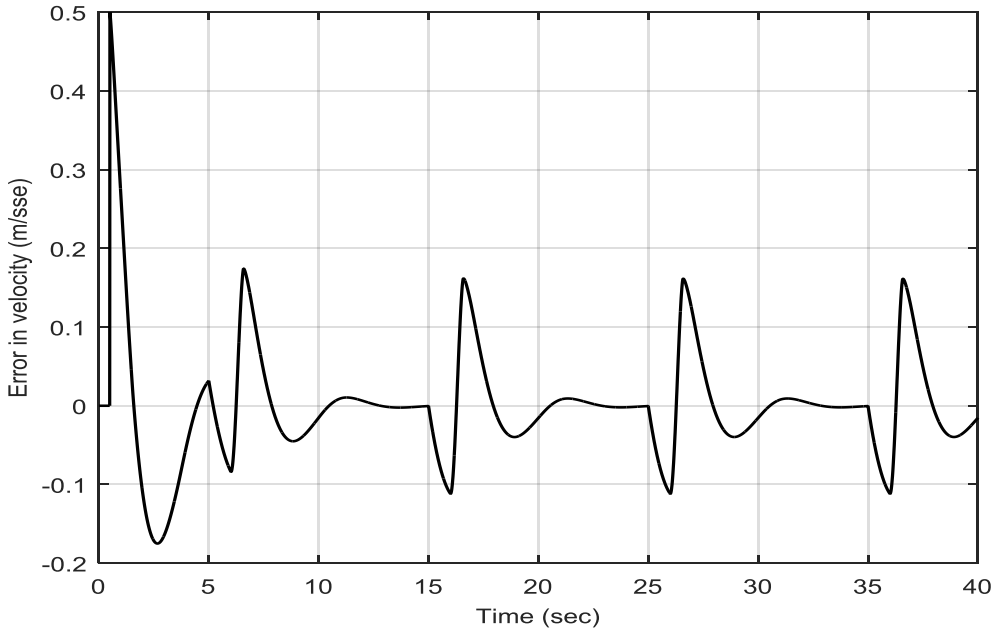



Fig. 3.43 Error in velocity for lemniscate trajectory using PID controller.

Moreover, the trajectory tracking for both X and Y coordinates of the square trajectory is given in Fig. 3.44 and Fig. 3.45, respectively. The trajectory tracking errors of X and Y coordinates are depicted in Fig. 3.46. They demonstrate a sharp drop and rising alongside to both coordinates. The maximum X-coordinate error in the square trajectory is equal to '1m' whilst the minimum X-coordinate error is equal '-1.25m'. Correspondingly, the maximum Y-coordinate error in the square trajectory is equal to '1.8m' whilst the minimum Y-coordinate

error is equal '0.3m'. In fact, this significant value of tracking error reflects that the PID control is not an efficient approach for a non-continuous gradient trajectory. This might be solved if the model of the UGV is being simplified and hence this will imply ramifications on the performance of the UGV. Therefore, a development of a new control strategy is the effective solution to improve the operation of the UGV and minimise the tracking error greatly.

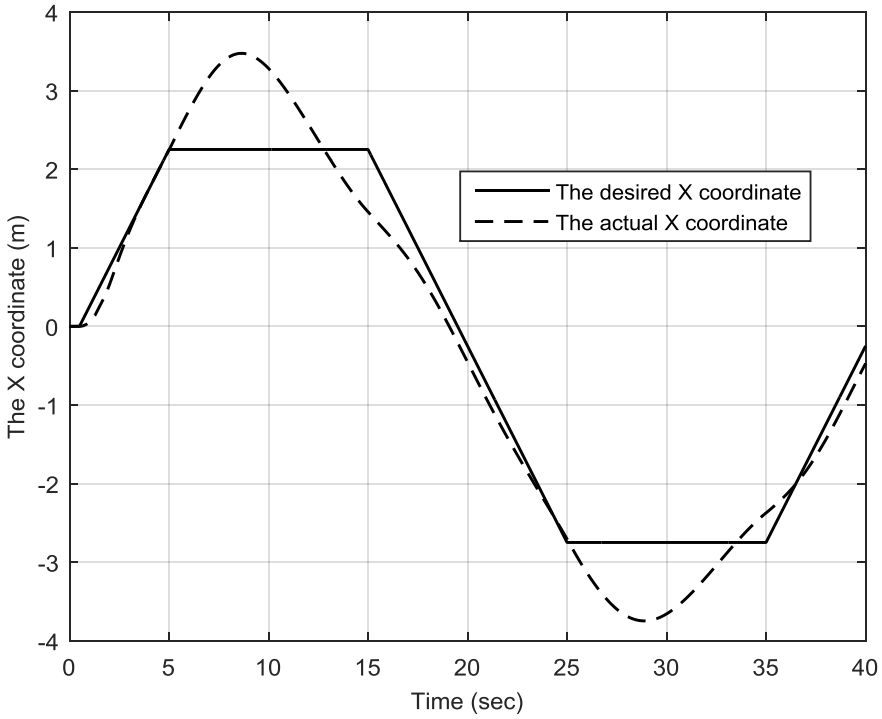



Fig. 3.44 X coordinates for square trajectory using PID controller.

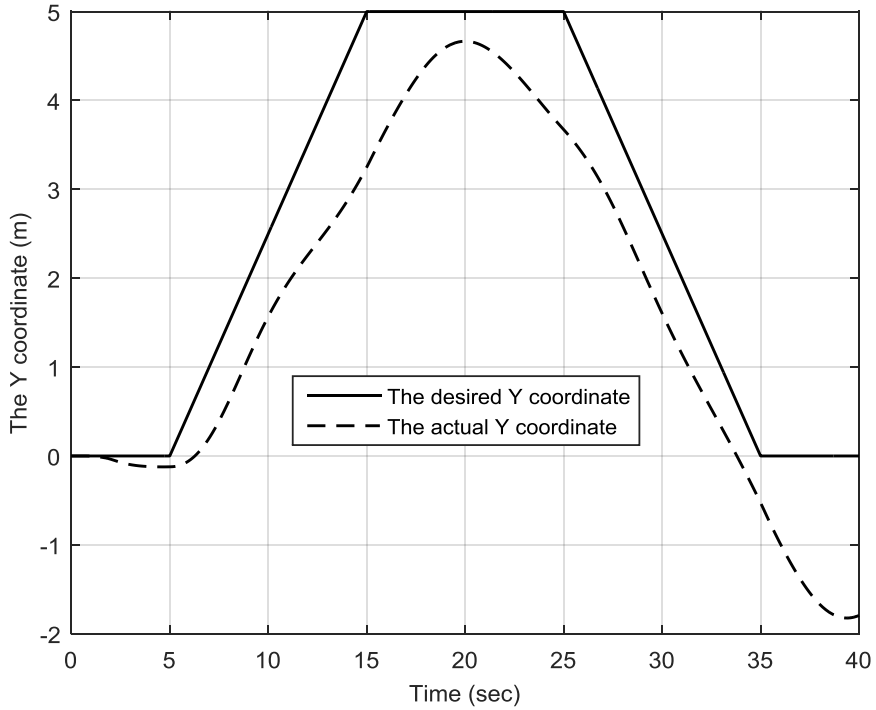



Fig. 3.45 Y coordinates for square trajectory using PID controller.

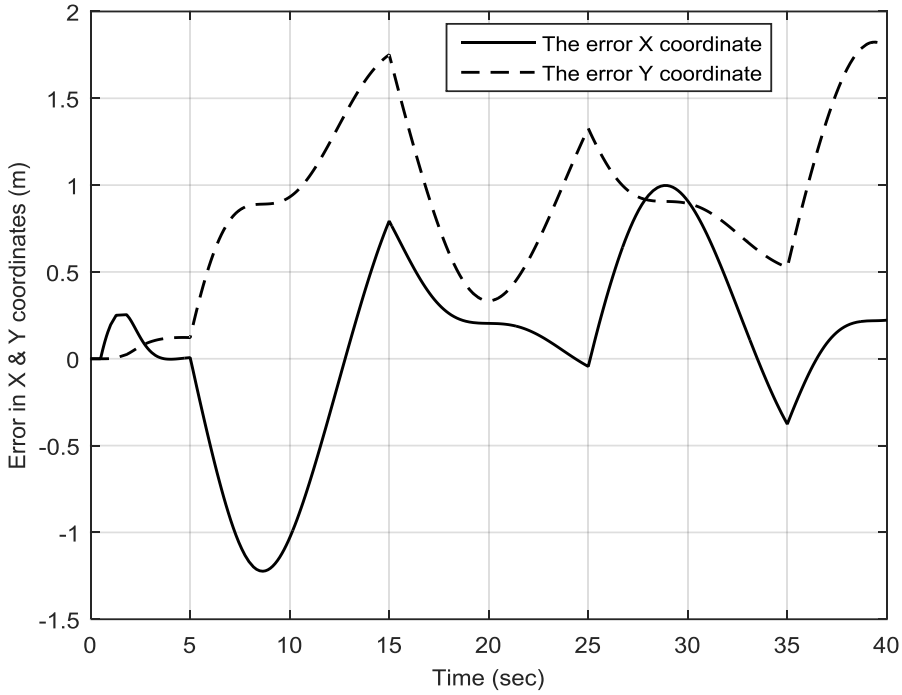



Fig. 3.46 Error in X and Y coordinates for square trajectory using PID controller.

## 3.7 Chapter Summary

A description of locomotion of an unmanned ground vehicle is introduced. This involves the definition of the two main characteristics, i.e. holonomic and non-holonomic. It implies the study of some constraints that emerges based on the fact of existing of non-holonomic characteristic. Then, the modelling of the UGV is presented. The modelling has involved three main components based on kinematic, dynamic and actuating characteristics. Each component is thoroughly investigated and modelled based on its governing equations. A traditional PID controller is implemented based on the literature. The simulation results are conducted based on four trajectories that comprise continuous and non-continuous gradient routes. Although the simulation results demonstrate a reasonable performance for the continuous gradient routes, the trajectory-tracking error has been still highly observed in the non-continuous gradient square trajectory. This reveals a strong demand for proposing a new control methodology to improve trajectory-tracking error largely. In the next chapter, a novel control strategy based on a fractional PID controller is conducted to minimise trajectory-tracking error and improve control efforts. Trajectory tracking will be investigated thoroughly based the same trajectories to highlight the main differences and similarities between the conventional and fractional order PID controllers.

# Chapter 4

# Trajectory Tracking of UGV Based on Fractional Order PID Controller

## 4.1 Introduction

Fractional order of a proportional integral derivative controller is increasingly becoming a popular technique that has gained an increasing attention in recent years in industrial applications. In the state of the art, many applications have been reported. Aboelela et al. (2012) designed a fractional order proportional integral derivative (FOPID) to control a trajectory of a flight path of six degrees of freedom. Aldair and Wang (2010) introduced an optimal fractional order PID controller for a full vehicle nonlinear active suspension system. The optimal values of the fractional PID controller's parameters are tuned using an evolutionary algorithm. Zamani et al. (2009) presented an application of a fractional order PID controller to an automatic voltage regulator. The controller employed a particle swarm optimization algorithm to carry out the design procedure of the fractional order PID controller.

The fractional order PID controller has structure based parameters. It has been proved that its performance is robust for uncertainties in robotic systems. It is apparent that the fractional order PID controller would involve many optimization algorithms based on evolutionary and swarm intelligence for tuning and obtaining optimal parameters. Lee and Chang (2010) presented two optimization algorithms for optimizing a fractional order PID controller. These algorithms are an electromagnetism and evolutionary algorithm. Hence, a combination of two algorithms was introduced to take the advantages of both algorithms and reduce the computation complexity of the electromagnetism algorithm.

Cao et al. (2005) introduced an intelligent optimization method for designing a fractional order controller based on a genetic algorithm. The optimization of a designing process was analysed for obtaining of proportional, integral, derivative, derivative order and integral order based on the genetic algorithm. Furthermore, a PSO has been regarded widely as a promising optimization algorithm due to its combination of simplicity (in terms of its implementation), low computational cost and good performance. Concomitant, optimal problems solved by genetic algorithms can be obtained better solutions with PSO in comparison with conventional methods. These are precisely the main motivations that the PSO will be applied for designing

the fractional PID controller design to obtain the optimal parameters (Cao and Cao, 2006; Ramezanian and Balochian, 2013).

Accordingly, this chapter investigates a trajectory tracking problem of unmanned ground vehicles using a proposed fractional order PID controller. The controller's parameters are optimized by using a particle swarm algorithm. The algorithm is utilized for searching the optimal values for the controller's parameters. These parameters are pivotal in designing the fractional order PID controller. The models of kinematics, dynamics and actuators are used to implement an accurate mathematical representation and thus a robust fractional order PID is required. The used tuning approach for the PSO algorithm is based on the integral square of error (ISE) method.

## 4.2 Chapter Organisation

The chapter is organised as follows: Section 4.3 provides a theoretical overview of fractional order systems. It has the structure of the fractional order PID controller that is introduced to tackle the trajectory tracking problem. The particle optimisation algorithm is described in Section 4.4. In Section 4.5, the control system design is discussed. The stability of the system is analysed in Section 4.6. The simulation results are conducted in Section 4.7. The robustness of the system is investigated in Section 4.8. Finally, the chapter summary is given in Section 4.9.

## 4.3 Fractional Order Systems

In recent years, researchers have utilised fractional order systems for modelling various structures. The fractional order systems have demonstrated better responses than traditional controller systems. They have shown several advantages such as increasing speed of the response, decreasing the steady state error and improving the stability of control systems (Monje et al., 2010).

### *4.3.1 Fractional Order Calculus*

Fractional calculus is a mathematical topic which studies the ability of taking real number power of both the differential and integration operators. There are several definitions to describe the fractional derivative. The firmly established definitions are Grunwald–Letnikov definition and the Riemann– Liouville definition. The most frequently used definition in fractional-order calculus is the Riemann–Liouville definition (Podlubny, 1999), in which the fractional order integration is defined as follows:

$$ {}_{\alpha}D_t^{-\lambda} f(t) = \frac{1}{\Gamma(\lambda)} \int_{\alpha}^{t} (t-\tau)^{\lambda-1} f(\tau) d\tau \tag{4.1} $$

where λ represents the real order of the differential and integral (0< λ <1); $\alpha$ is the initial time instance, often assumed to be zero; and 't' is the parameter for which both the differential and integral are accounted.

$\Gamma(\lambda)$ is Euler's gamma function

The Laplace and Fourier transforms of the fractional derivative of f (t) is given by:

$$ L\left[D_t^{\lambda} f(t)\right] = S^{\lambda} L[f(t)] - \sum_{k=1}^{n} S^{k} \left[D_t^{\lambda-k-1} f(t)\right]_{t=0} \tag{4.2} $$

For convenience, the second part on the right hand side of Equation (4.2) can be ignored when the derivatives of the function *f* (t) are all equal to '0' at t=0. Therefore, this equation can be rewritten as in below:

$$ L\left[D_t^{\lambda} f(t)\right] = S^{\lambda} F(s) \tag{4.3} $$

where F(s) is the Laplace transformer of *f*(t).

### *4.3.2 Fractional Order PID Controller*

The integral-differential equation defining the control action of a FOPID controller is given by:

$$ u(t) = K_p e(t) + K_i D^{-\lambda} e(t) + K_d D^{\mu} e(t) \tag{4.4} $$

Applying Laplace transform to this equation with null initial conditions, the transfer function of the controller can be expressed by:

$$ C_f(s) = K_p + \frac{K_i}{S^{\lambda}} + K_d S^{\mu} \tag{4.5} $$

In a graphical way, the control possibilities using a fractional-order PID controller are shown in Fig. 4.1, extending the four control points of the classical PID to the range of control points of the quarter-plane defined by selecting the values of λ and μ.

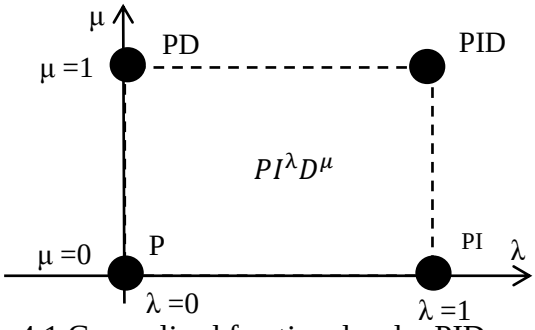



Fig. 4.1 Generalised fractional order PID controller.

The essential advantage of the fractional order PID controller is the less sensitive to changes might happen to parameters of a controlled plant. In fact, the two extra degrees produce more adjustment for the dynamic behaviour of the fractional order PID controller than a conventional case. The Simulink block diagram configuration of a fractional order PID controller is depicted in Fig. 4.2. A fractional PID controller uses a fractional derivative and a fractional integral. The transfer functions of these operations are $s^{\mu}$ and $1/s^{\lambda}$ where $\mu$ and $\lambda$, which are both unity in a PID controller are not integers.

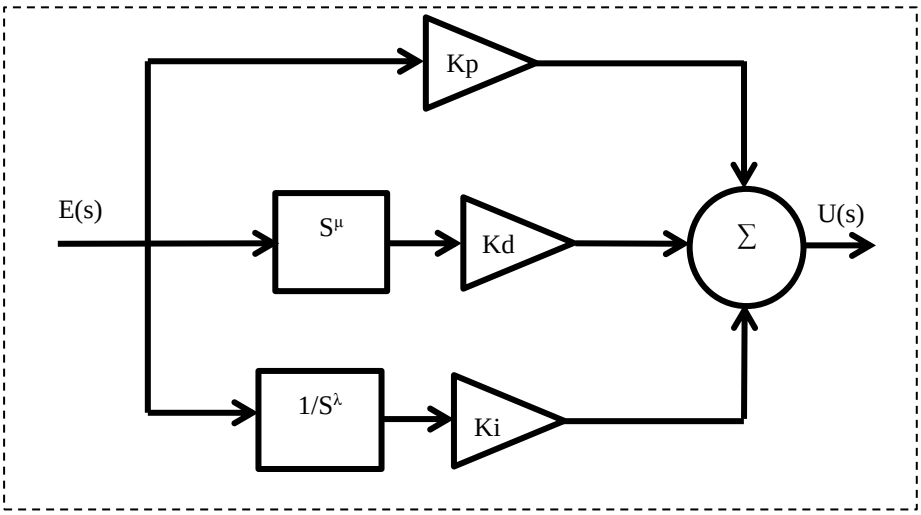



Fig. 4.2 Block diagram of a fractional order PID.

## 4.4 Particle Swarm Optimization

Traditionally, parameters of a system have been obtained from the trial and error approach, which consumes a large amount of time in reaching the best choice of gains. To reduce a complexity of the traditional approach, a particle swarm optimization is used to solve a wide range of practical problems including optimization and designing gains of fractional order PID controllers. It is a computational method that optimizes a problem by iteratively trying to improve a candidate solution about a given measurement of a quality. It can obtain suboptimal solutions for difficult problems when conventional methods fail to produce in a reasonable time.

Evolutionary algorithms (EA) can be a useful paradigm and provide promising results for solving complex optimization functions. Evolutionary computation refers to the study of computational systems that use ideas to draw inspirations from natural evolution. Evolutionary algorithms such as genetic algorithm, simulated annealing and ant colony optimization, have been extensively employed in control applications to efficiently search global optimum

solutions. Nonetheless, a particle swarm optimisation (PSO) draws more attention than other algorithms due to its simplicity and efficiency.

The PSO is a stochastic algorithm based on the principle of a natural selection and a search algorithm. This algorithm is inspired by the study of birds and fish flocking. In such an algorithm, each particle in a swarm represents a solution to a problem and it is defined within its position and velocity. In a D-dimensional search space, a position of an $i^{th}$ particle can be represented by a D-dimensional vector, $S_i = (X_{i1,\ldots}, X_{iD})$. The velocity of a particle $V_i$ can be represented by another D-dimensional vector $V_i = (V_{i1,\ldots}, V_{iD})$. The best position visited by the $i^{th}$ particle is denoted as $pbest_i = (P_{i1}, \ldots, P_{iD})$, and $gbest$ is denoted an index of a particle that visited the best position in a group of swarm, thus, $gbest$ becomes the best solution found so far (Sadati et al., 2006; Zhang et al., 2013).

The principle steps of the PSO algorithm are illustrated as in the following procedures:

1- Initialize
   a) Set constants $W$, $c_1$, $c_2$.
   b) Randomly initialize particle positions $x_0^i \in D$ for $i = 1, \ldots., p$.
   c) Randomly initialize particle velocities $0 \leq v_0^i \leq v_0^{max}$ for $i = 1, \ldots., p$.
   d) Set k=1.
2- Optimize
   a) Evaluate function value $f_k^i$ using design space coordinates $x_k^i$.
   b) If fitness function $f_k^i \leq f_{best}^i$ then $f_{best}^i = f_k^i, p_k^i = x_k^i$.
   c) If fitness function $f_k^i \leq f_{best}^i$ then $f_{best}^i = f_k^i, p_k^i = x_k^i$.
   d) If stopping condition is satisfied then go to 3.
   e) Update all particle velocities $v_k^i$ for $i = 1, \ldots., p$.
   f) Update all particle positions $x_k^i$ for $i = 1, \ldots., p$.
   g) Increment k.
   h) Go to 2(a)
3- Terminate

The embedded of the PSO algorithm with the fractional order PID controller is described in the following steps:

**Step 1:** The algorithm parameters such as a number of generations, swarm size, inertia weight, and maximum iterations are initialized.

$$W = W_{max} - \frac{(W_{max} - W_{min})}{iter_{max}} \text{ . iter} \tag{4.6}$$

where $W$ is the inertia weight.

**Step 2:** The values of $K_p$, $K_i$, $K_d$, $\lambda$ and $\mu$ for each controller are initialized randomly within the optimal range of values for each parameter.

**Step 3:** The fitness of each particle is evaluated using an integration square of an error (ISE). The fitness function is given as in Equation (4.7).

$$\text{f(t)} = (\int_0^\infty [e(t)]^2 dt) \tag{4.7}$$

**Step 4:** The local best position ($pbest_i$) and the global best position ($gbest$) of particles are obtained based on the fitness value calculated from **step 3** for each particle.

**Step 5:** In each iteration, the velocity and position of the particle is updated using the Equation (4.8) and Equation (4.9) respectively.

$$V_i^{k+1} = WV_i^k + C_1 R_1 \left(pbest_i - S_i^k\right) + C_2 R_2 \left(gbest_i - S_i^k\right) \tag{4.8}$$

$$S_i^{k+1} = S_i^k + V_i^{k+1} \tag{4.9}$$

where $R_1$ and $R_2$ are random numbers selected between '0' and '1'.

$c_1$and $c_2$ are the acceleration constants which influence the convergence speed of each particle.

**Step 6:** The steps from '2' to '5' are repeated until the maximum iterations reached or the best solution is obtained.

The flow chart for particle swarm optimization is given Fig. 4.3 below:

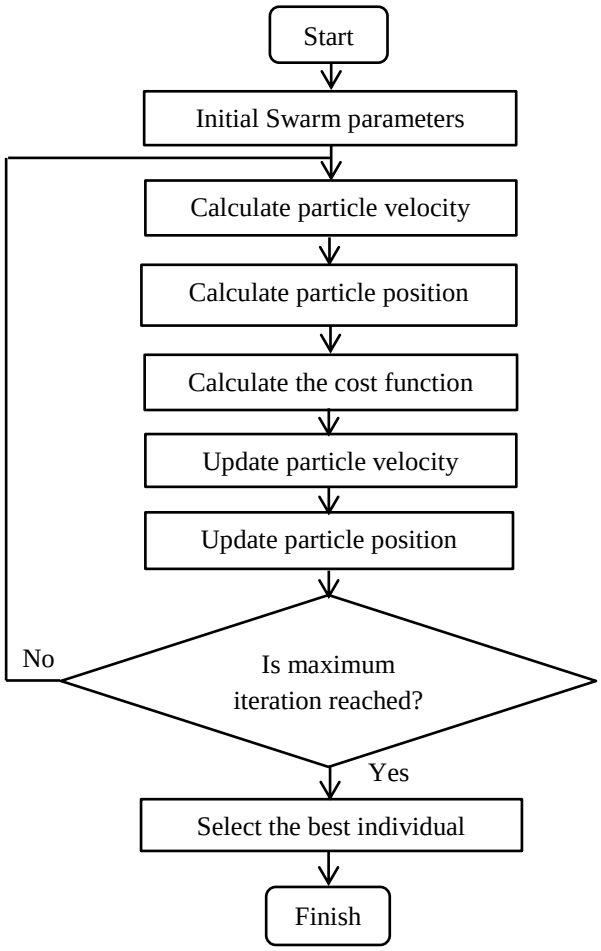



Fig. 4.3 Structure of a particle swarm optimization.

The basic variants that have used in the PSO algorithm and their definitions are tabulated in Table 4.1.

Table 4.1 Definitions of the PSO variables.

| Variable | Definition |
|---|---|
| k | Iteration number |
| Iter | The current number of iterations |
| $Iter_{max}$ | Maximum number of iterations |
| D | Dimension search space |
| $S_i^k$ | The current position of particle $i^{th}$ at iteration k |
| $V_i^k$ | The current velocity of particle $i^{th}$ at iteration k |
| $pbest_i$ | Local best position y visited by $i^{th}$ particle |
| $gbest$ | Global best position visited by a warm |

| | |
|---|---|
| W | Inertia weight function |
| $W_{max}$ | Maximum value of inertia weight |
| $W_{min}$ | Minimum value of inertia weight |
| $C_1$ | Cognitive coefficient |
| $C_2$ | Social coefficient |
| $R_1$ & $R_2$ | Random number between 0 and 1 |

## 4.5 Control System Design

Two fractional order PID controllers are proposed for driving the wheels of an unmanned ground vehicle separately. The block diagram of the proposed fractional order PID controllers integrated with the model of the UGV is depicted in Fig. 4.4. The input of the first controller is the difference between the desired generated trajectory and actual instantaneous trajectory. Therefore, the vehicle must change its orientation frequently to track the desired trajectory. The output is for controlling the right wheel. The second controller receives the difference between a desired and an actual velocity as reference input. The desired velocity was represented by a constant value. The outputs of both controllers are fed to the actuators of the unamnned ground vehicle for controlling the motion and tracking the given trajectories.

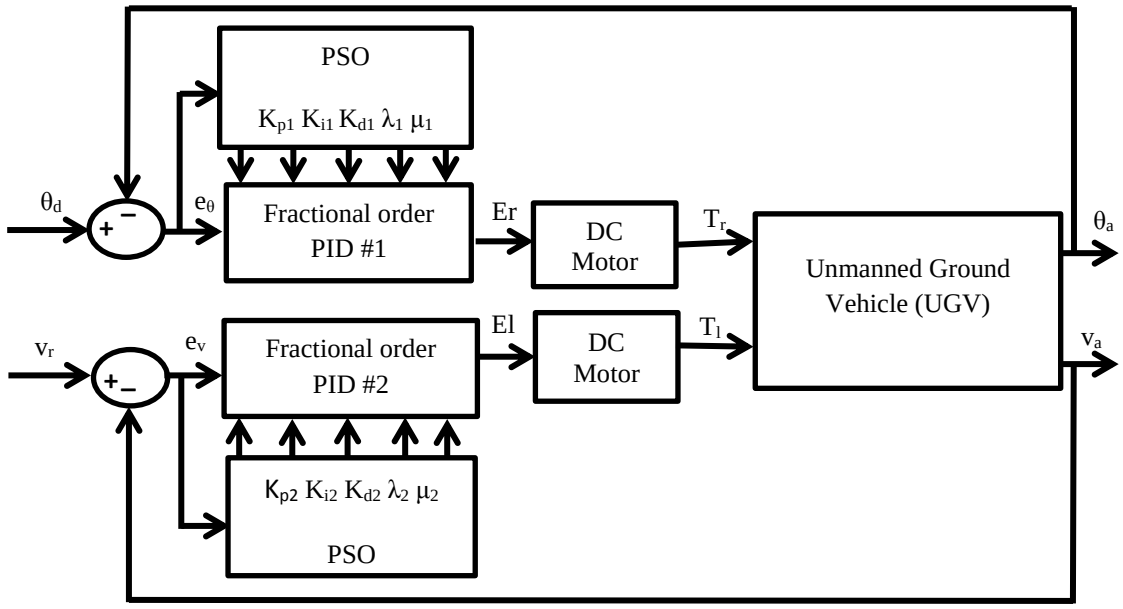



Fig. 4.4 Block diagram of the fractional order PID controller and UGV.

The cost function used in PSO algorithm is simulated by using the integral square of the error method for both the orientation and velocity as shown in Equations (4.10) and (4.11).

$$ISE\ \#1 = \left( \int_0^\infty [e_\theta(t)]^2 dt \right) \tag{4.10}$$

$$ISE\ \#2 = \left( \int_0^\infty [e_v(t)]^2 dt \right) \tag{4.11}$$

The orientation error and velocity error signals of the first and second fractional order PID controllers are given in equations below, respectively:

$$e_\theta = \theta_d - \theta_a \tag{4.12}$$

$$e_v = v_d - v_a \tag{4.13}$$

where

$\theta_d$ - desired orientation,

$\theta_a$ - actual orientation,

$e_\theta$ - orientation error,

$v_d$ - desired velocity,

$v_a$ - actual velocity,

$e_v$ - velocity error.

In our experiments, the PSO algorithm has used the following values for its initial variants i.e. $W_{max}$=1, $W_{min}$=0.25, $Iter_{max}$=5, D=10, $c_1 = c_2$ =1.4, the dimension of the problem is D=10, the size of the swarm equals '5'. The optimal values for the first and second controllers are given in Table 4.2 and Table 4.3 respectively.

Table 4.2 Parameters of the first fractional order PID controller.

| Parameters | $K_{p1}$ | $K_{i1}$ | $K_{d1}$ | $\lambda_1$ | $\mu_1$ |
|---|---|---|---|---|---|
| Optimal Value | 8.58 | 0.36 | 10.71 | 1.12 | 0.5 |

Table 4.3 Parameters of the second fractional order PID controller.

| Parameters | $K_{p2}$ | $K_{i2}$ | $K_{d2}$ | $\lambda_2$ | $\mu_2$ |
|---|---|---|---|---|---|
| Optimal Value | 3.35 | 4.33 | 1.23 | 0.89 | 1.10 |

From the two tables above, the fractional order transfer functions of the first and second fractional order PID controllers are given below respectively:

$$F_1(s) = \frac{1.23s^{1.99} + 3.35s^{0.89} + 4.33}{s^{0.89}}$$

$$F_2(s) = \frac{10.71s^{1.62} + 8.58s^{1.12} + 0.36}{s^{1.12}}$$

## 4.6 Stability Analysis

In this section, the stability analysis of the proposed system will be investigated. Different methods can be applied to test the stability of a given system such as using Nyquist stability criterion, which is based on Cauchy's theorem concerned with mapping contours in a complex S-plane. The Nyquist stability criterion states that, a system is asymptotically stable if all its poles are placed in the left hand plane and the Nyquist diagram does not enclose the point '-1'. Vivero and Liceaga-Castro (2008) proposed a multi-input multi-output (MIMO) MATLAB toolbox for analysing the stability of multivariable systems. However, this toolbox cannot be applied for fractional order systems because it deals with continuous systems only. Therefore, this will be manipulated based on analysing the continuous-time transfer function of each part of the MIMO system. Then, this will be integrated with a fractional transfer function of each control to obtain the overall fractional transfer function. Consequently, the MIMO system is connected with two diagonal fractional order controllers. The interconnection between the MIMO loops with fractional order controllers is shown in Fig. 4.5.

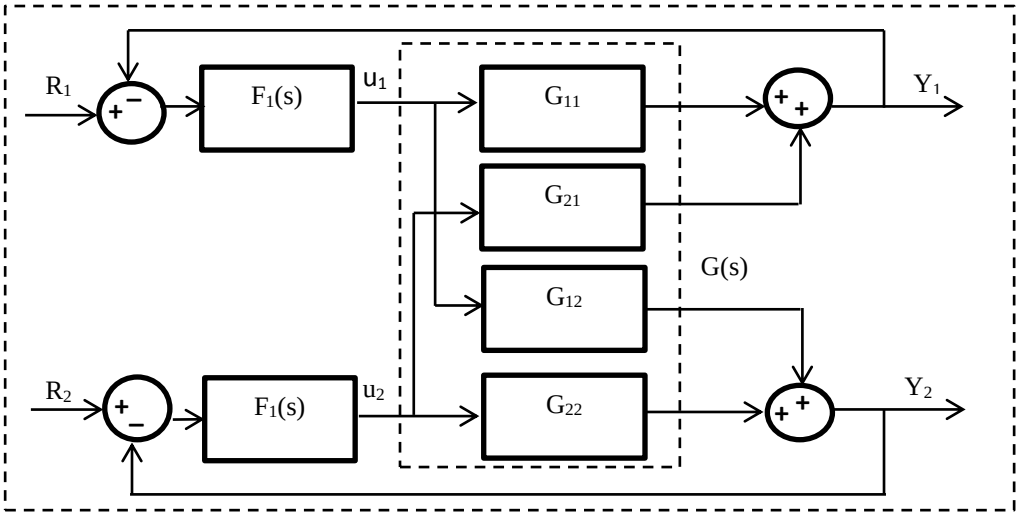



Fig. 4.5 Multivariable system with two controllers.

$F_1(s)$ and $F_2(s)$ are the two designed fractional order PID controllers. $G_{11}$, $G_{12}$, $G_{21}$ and $G_{22}$ are the transfer functions of the system. These transfer functions are obtained from the equations that driven based on the modelling of the unmanned ground vehicle. The mathematical model who derived in Chapter 3 is implemented based on a state space model. This can be easily transformed to transfer function representations using continuous time systems in MATLAB. The framework of the total closed loop transfer function is obtained from a multivariable system based on designing individual channels (Ugalde-loo et al., 2005). This is decomposed into an equivalent set of Single-Input Single-Output (SISO) systems. Each SISO system is an open loop channel that transmits between output $Y_i(s)$ and Input $R_i(s)$ with all internal loops.

The multivariable system G(s) with 2x2 systems and the controllers are $F_i(s)=diag[F_1(s), F_2(s)]$. The transfer function for each individual channel can be established as in the following relationships:

$$C_i(s) = \frac{Y_i(s)}{R_i(s)} \quad (i = 1,2) \tag{4.14}$$

where $C_1(s)$ *is* the first individual channel that can be defined in the following equation:

$$C_1(s) = \frac{Y_1(s)}{R_1(s)} = F_1(s)G_{11}(s)(1 - \gamma(s)H_2(s)) \tag{4.15}$$

Similarly, for the second individual channel $C_2(s)$ is given below;

$$C_2(s) = \frac{Y_2(s)}{R_2(s)} = F_2(s)G_{22}(s)(1 - \gamma(s)H_1(s)) \tag{4.16}$$

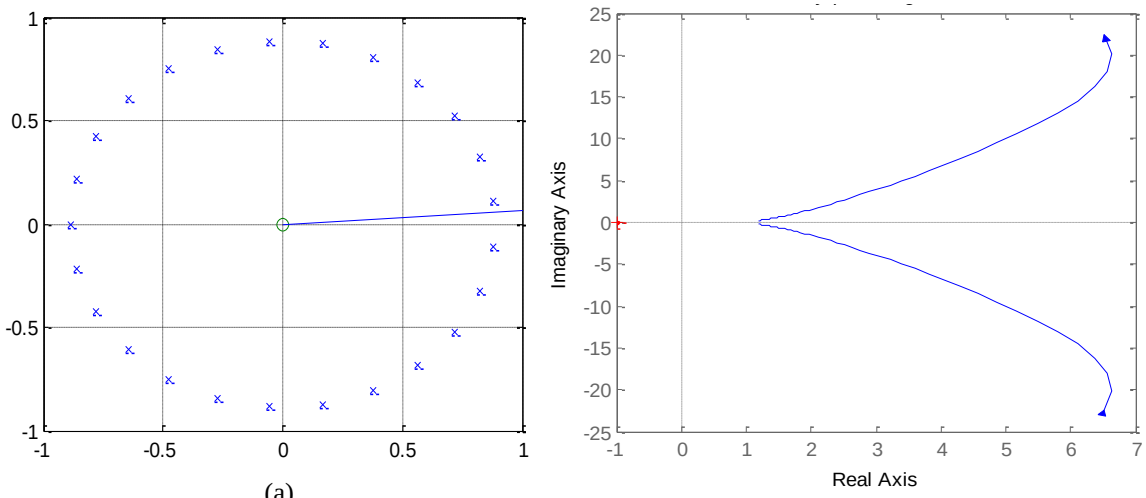
where $\gamma(s)$ is the multivariable structure function and $H_i(s)$ is a unity negative feedback subsystem, which are respectively defined in equations below:

$$\gamma(s) = \frac{G_{12}(s)G_{21}(s)}{G_{11}(s)G_{22}(s)} \tag{4.17}$$

$$H_i(s) = \frac{F_i(s)G_{ii}(s)}{1 + F_i(s)G_{ii}(s)} \ (i = 1,2) \tag{4.18}$$

The stability assessment for a single input single output (SISO) of a fractional order transfer function for the first individual channel '$C_1(s)$' was carried out based on MATLAB function written in reference (Chen et al., 2009). In this function, a returned argument factor calls 'K' is used to determine the stability of the fractional order system, if K is '1' the system is stable and '0' for unstable. In addition, from the pole positions shown in Fig. 4.6(a), it is observable that all poles are placed with the stable region. Fig. 4.6(b) shows the Nyquist plot for the first individual channel. The stability can be assessed based on Nyquist stability criterion. It can be

noticed that Nyquist diagram does not enclose the point '-1', which confirms the stability of the system.




(a) (b)

Fig. 4.6 Stability of first individual channel (a) Pole positions; (b) Nyquist plot.

The frequency domain response can be obtained for fractional order transfer function by replacing the variable's' by $j\omega$. Fig. 4.7 depicts the frequency response for the SISO fractional order function of the first individual channel.

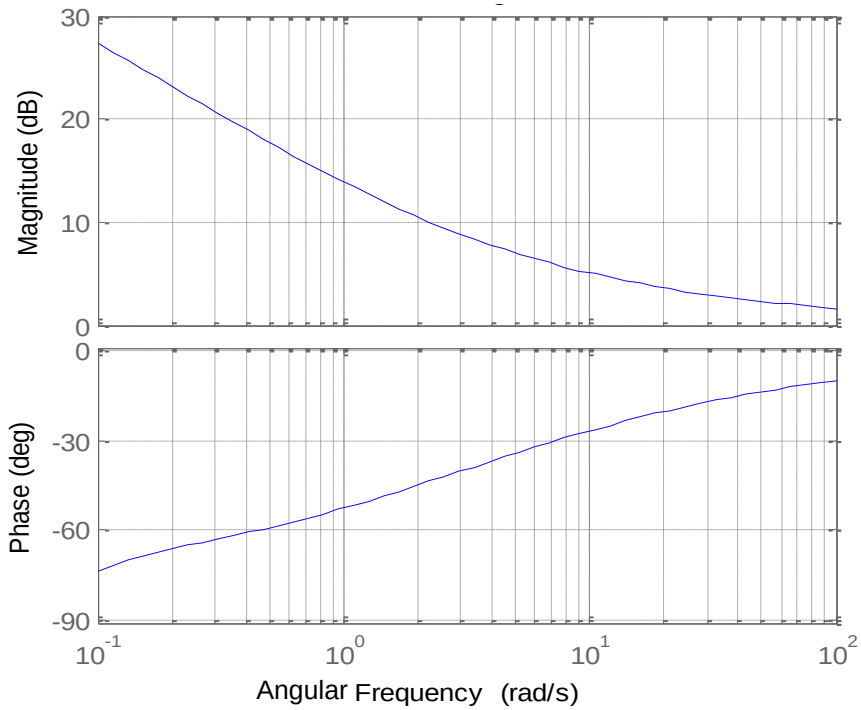
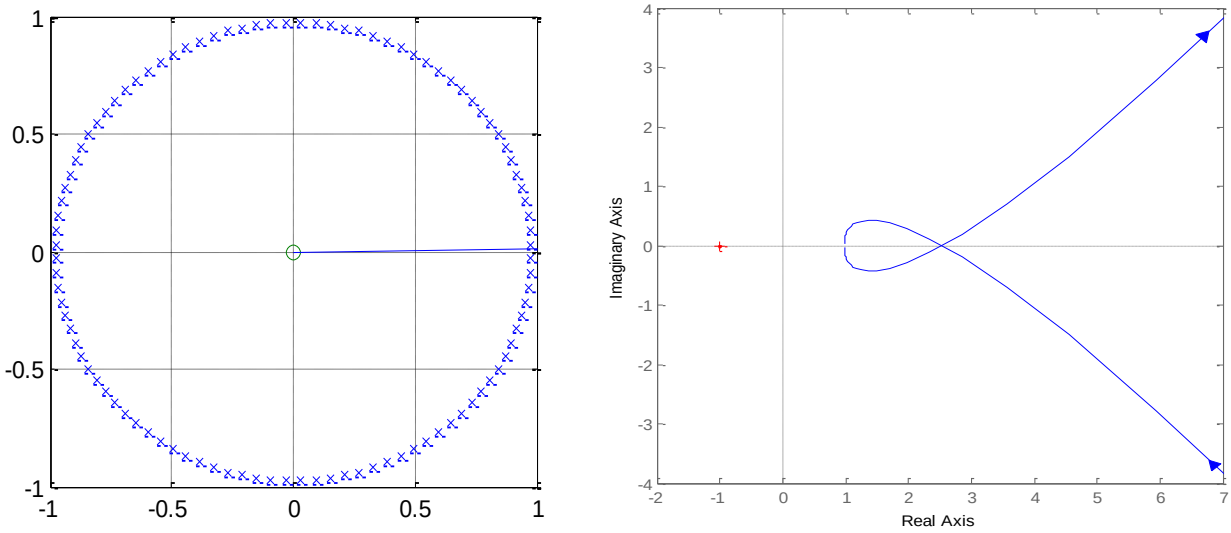



Fig. 4.7 Frequency-domain response for first individual channel.

Similarly, the stability assessment for SISO fractional order transfer function is performed for the second individual channel '$C_2(s)$'. In addition, the returned argument factor 'K' for '$C_2(s)$' is obtained when the simulation was executing. K indicates '1', which means the system, is stable. Fig. 4.8(a) demonstrates that all poles are placed in the stable region. Likewise, from Fig. 4.8(b), it can be noticed that Nyquist diagram does not enclose the point '-1'. Finally, the frequency domain response is introduced in Fig. 4.9.




(a) (b)

Fig. 4.8 Stability of second individual channel. (a) Pole positions; (b) Nyquist plot

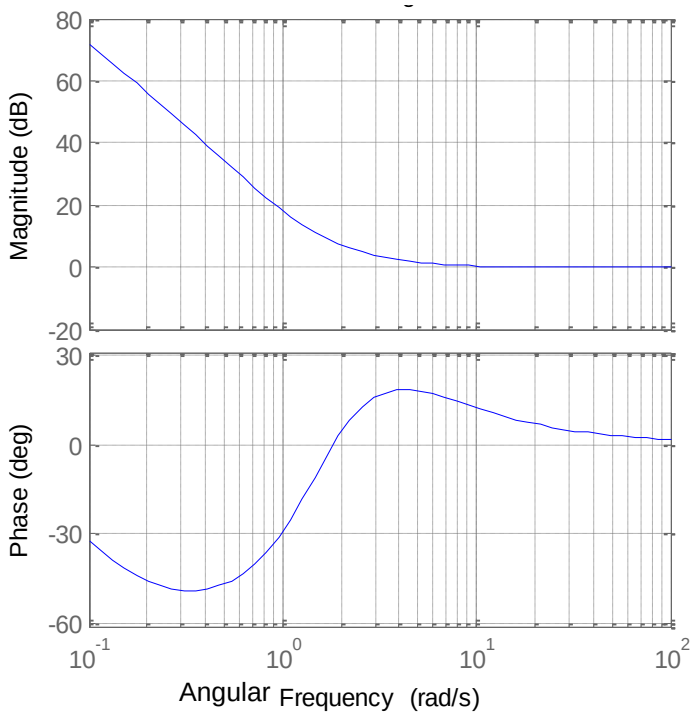



Fig. 4.9 Frequency-domain response for second individual channel.

## 4.7 Simulation Results

The simulation results are carried out to validate the proposed fractional order PID controllers using the four trajectories, i.e. linear, circular, lemniscate and square trajectories. The model unmanned ground vehicle is governed by the kinematic, dynamic and actuating equations given in Chapter 3. The entire system has two fractional order PID controllers and each controller has five parameters. These parameters are symbolized as $Kp_1$, $Ki_1$, $Kd_1$, $\lambda_1$ and $\mu_1$ for the first fractional order PID controller and similarly $Kp_2$, $Ki_2$, $Kd_1$, $\lambda_2$ and $\mu_2$ for the second fractional order PID controller. Particle swarm optimization is used to find the optimal values for those ten parameters. After the unmanned ground vehicle's model and the two fractional order PID controllers have been implemented, the platform is simulated and examined through different case studies to verify the performance of the proposed methodology. Each case study considers a different trajectory as follows:

### *4.7.1 Application Case 1*

In this case, a linear trajectory is generated based on a constant orientation i.e. $\theta_d = \pi/2$ $[rad]$ for interval $0 \leq t \leq 40$ $[sec]$. The simulation results of this case study are compared based on the traditional PID controlled presented in Chapter 3 and the proposed fractional order PID controller. The relationship between the desired trajectory and the actual trajectory based on each controller is shown in Fig. 4.10. The comparison demonstrates the effectiveness of the fractional order PID over the classic approach. The convergence of the actual trajectory based on the fractional order is feasible and minimise the trajectory tracking error. Fig. 4.11 demonstrates the UGV's orientation that is used for generating the linear trajectory. The error between the desired and the actual orientation is shown in Fig. 4.12 based on the conduced controllers. It has demonstrated a remarkable improvement compared with the traditional PID controller.

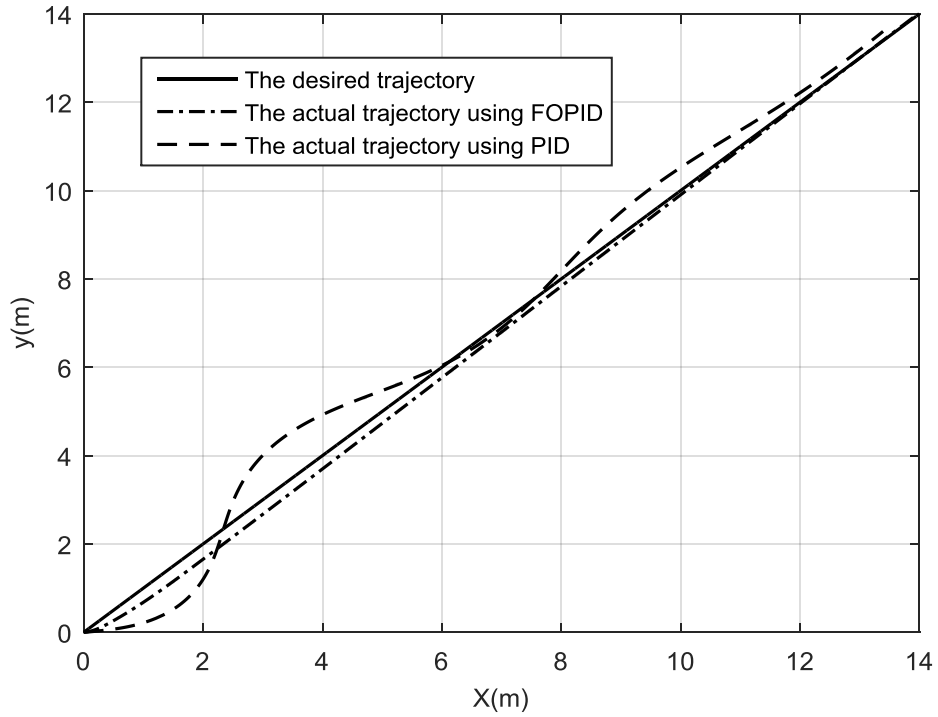



Fig. 4.10 Linear trajectories using PID and FOPID controllers.

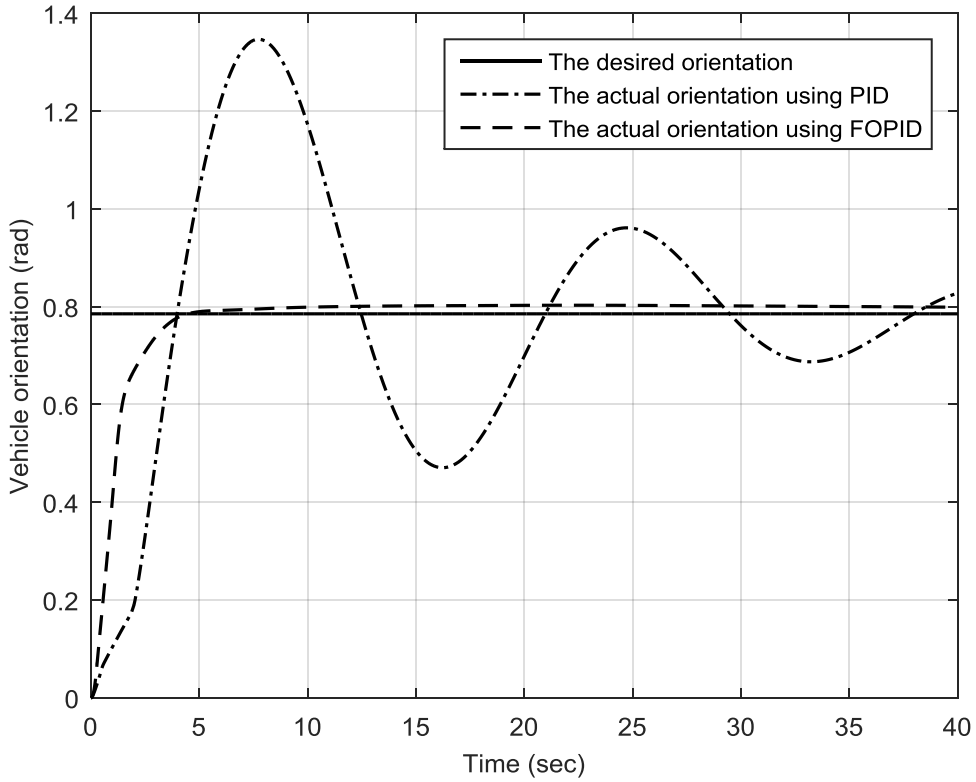



Fig. 4.11 Orientations for linear trajectory using PID and FOPID controllers.

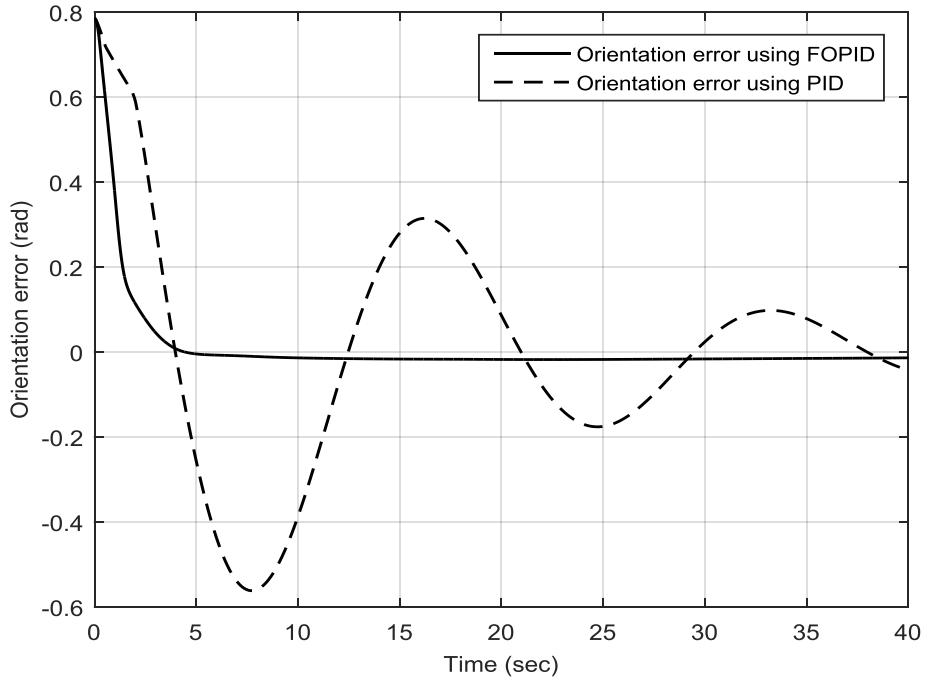



Fig. 4.12 Error in orientation for linear trajectory using PID and FOPID controllers.

Fig. 4.13 and 4.14 demonstrate the control actions needed from the left and right wheels to provide such a tracking response when both FOPID and conventional PID controller are used. It can be clearly observed that the control efforts needed in case of using FOPID are smoother, which validate a better design. Fig. 4.15 illustrates the ability and effectiveness of the second fractional order PID controller and the comparison with the conventional PID controller to track the desired velocity. The time response based on fractional order PID controller shows better steady state and overshoot. The obtained error between the desired and actual velocity using both PID and FOPID controllers is shown Fig. 4.16. This validates that improvement has occurred based on the proposed methodology. Figs. 4.17 to 4.20 show the trajectory tracking comparison for the coordinates of X-Y axes and the trajectory tracking errors. This confirms that the trajectory tracking has been improved significantly based on our design approach. Figs. 4.21 and 4.22 demonstrate the changing of the integral square error functions of both the orientation and velocity, respectively.

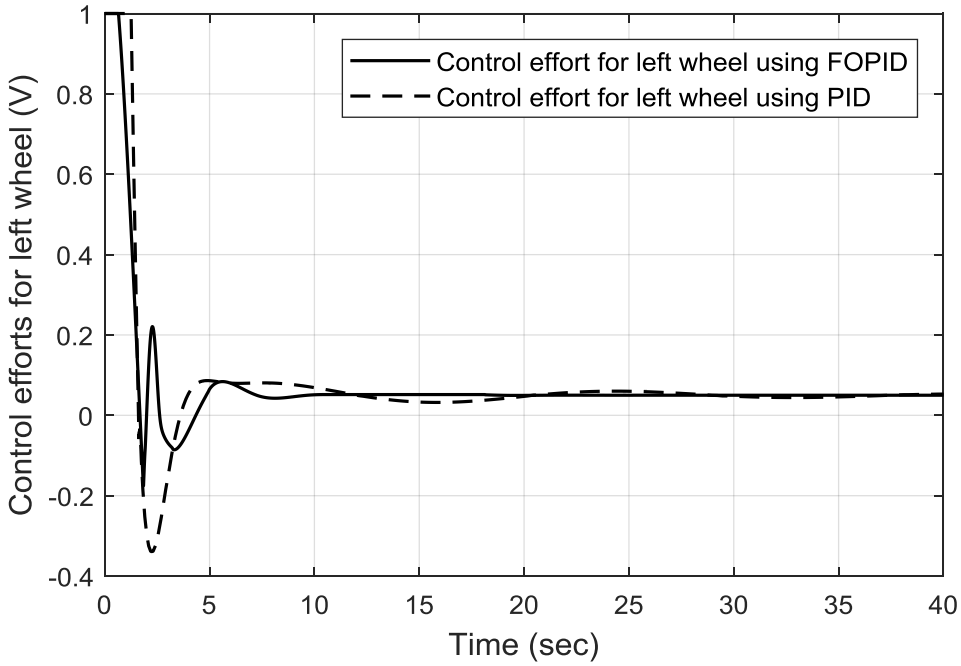



Fig. 4.13 Control efforts for left wheel of linear trajectory using PID and FOPID controllers.

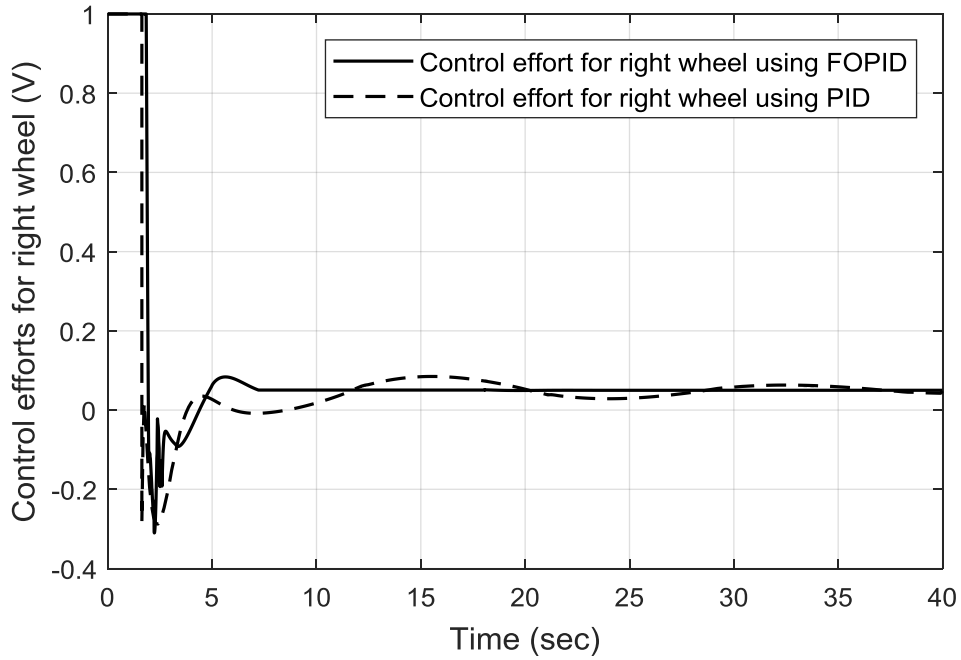



Fig. 4.14 Control efforts for right wheel of linear trajectory using PID and FOPID controllers.

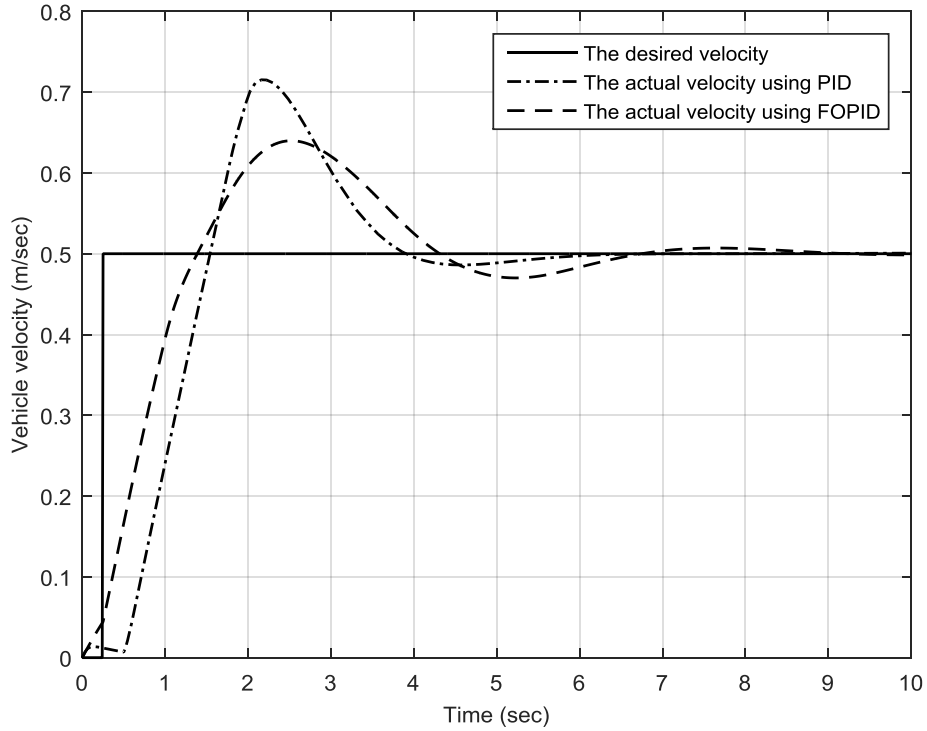



Fig. 4.15 Velocities for linear trajectory using PID and FOPID controllers.

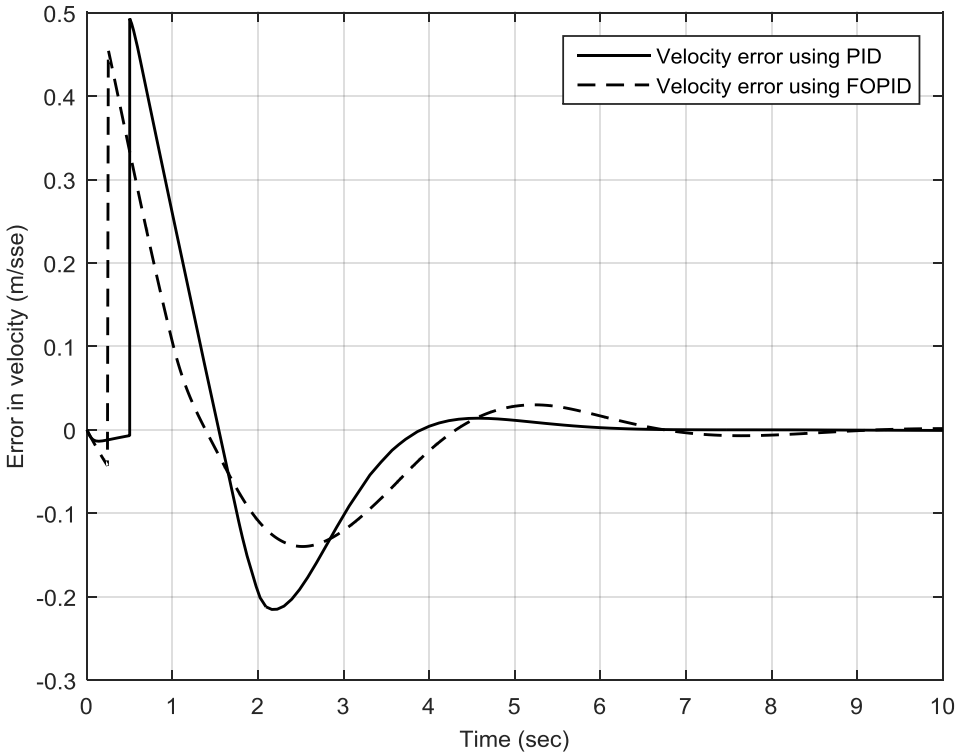



Fig. 4.16 Error in velocity for linear trajectory using PID and FOPID controllers.

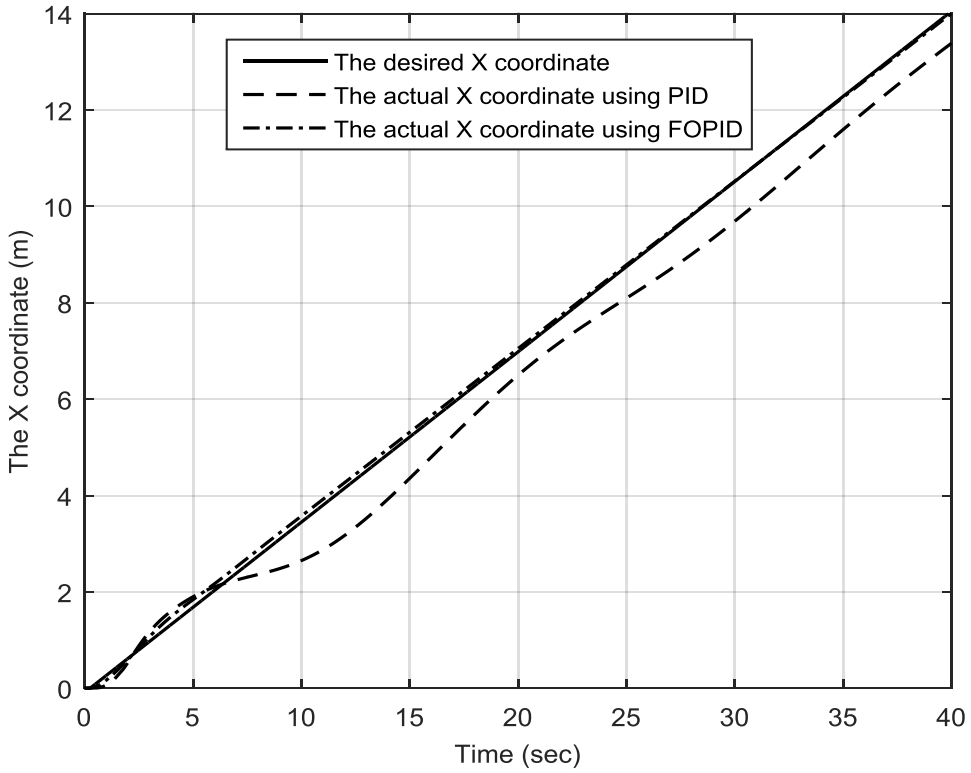



Fig. 4.17 X-coordinates for linear trajectory using PID and FOPID controllers.

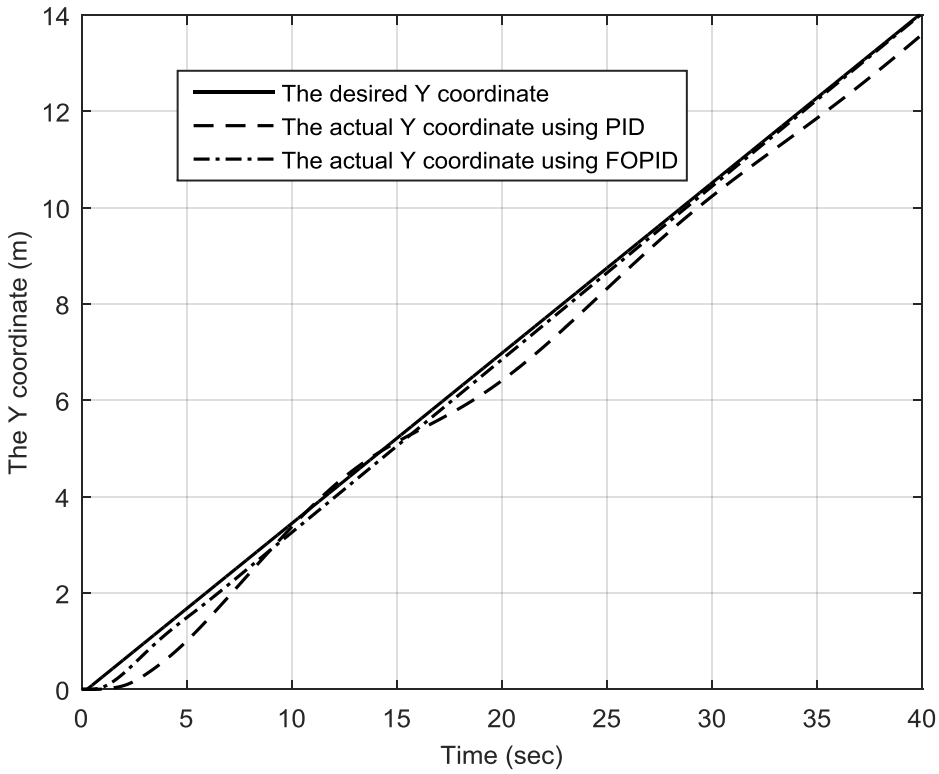



Fig. 4.18 Y-coordinates for linear trajectory PID and FOPID controllers.

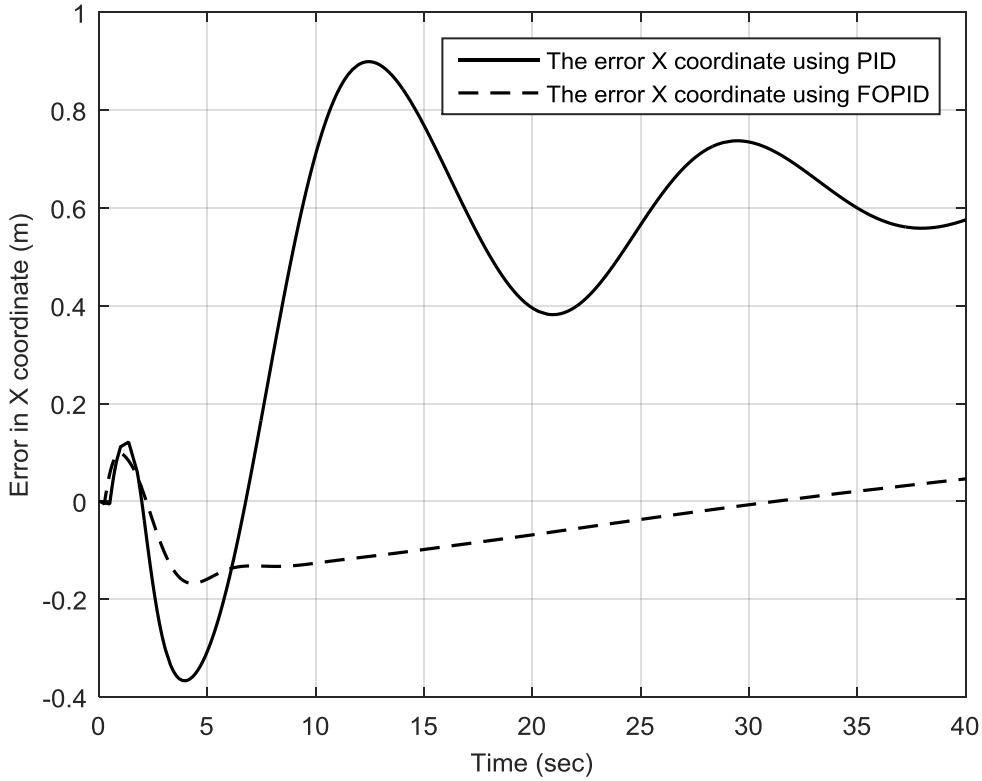



Fig. 4.19 Error in X coordinate for linear trajectory using PID and FOPID controllers.

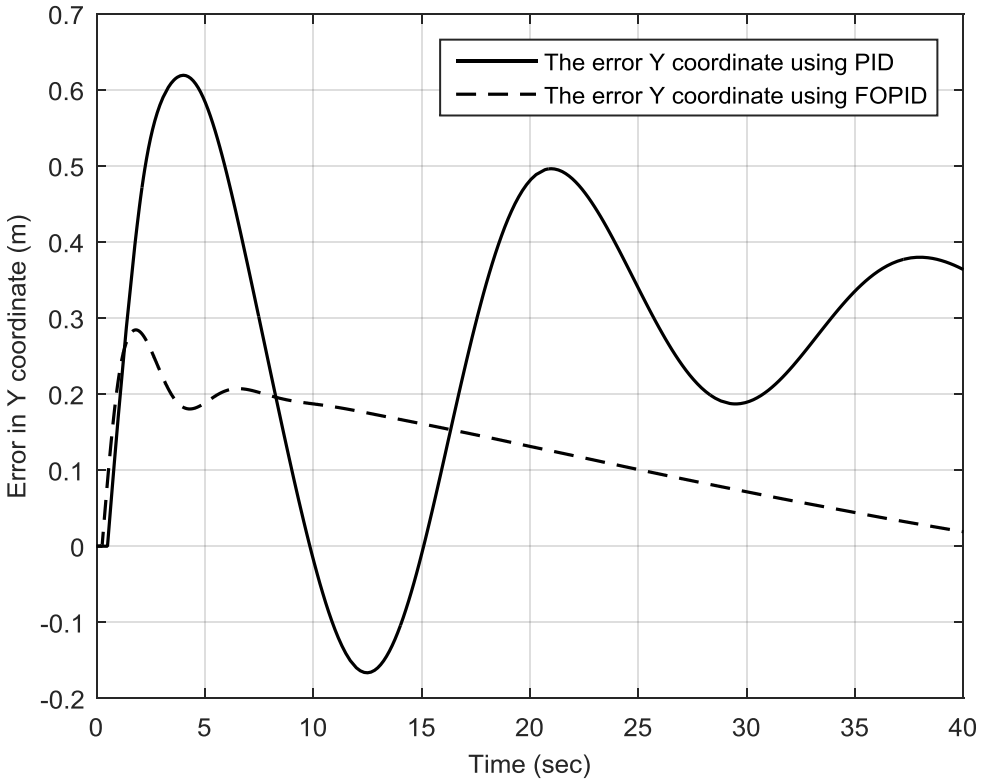



Fig. 4.20 Error in Y coordinate for linear trajectory using PID and FOPID controllers.

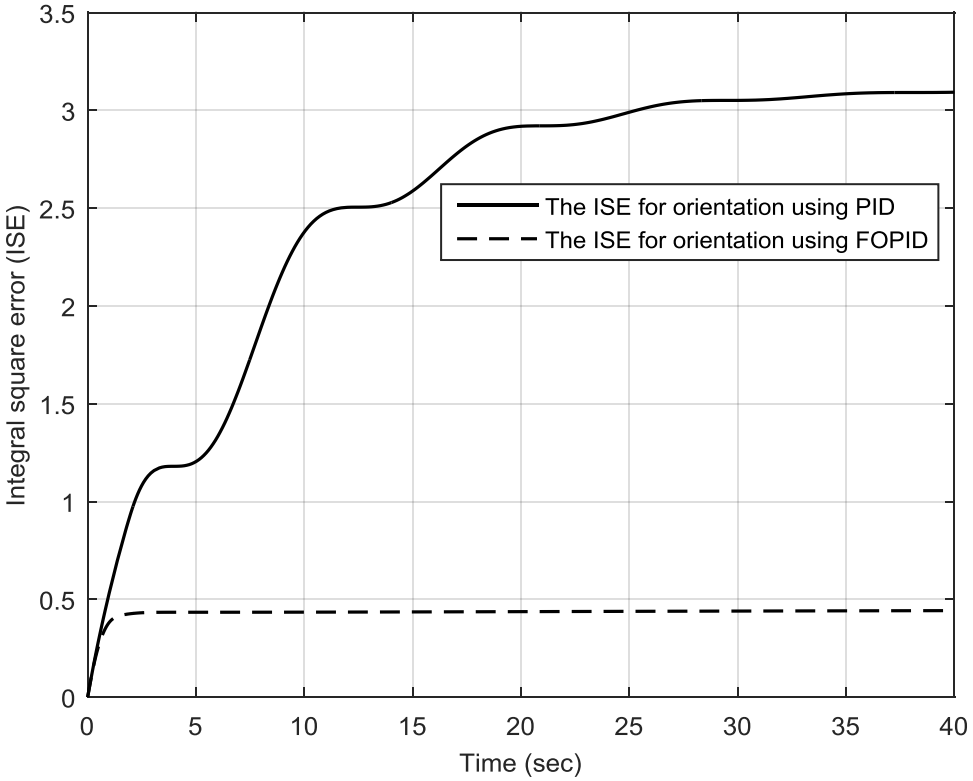



Fig. 4.21 ISE of the orientation for linear trajectory using PID and FOPID controllers.

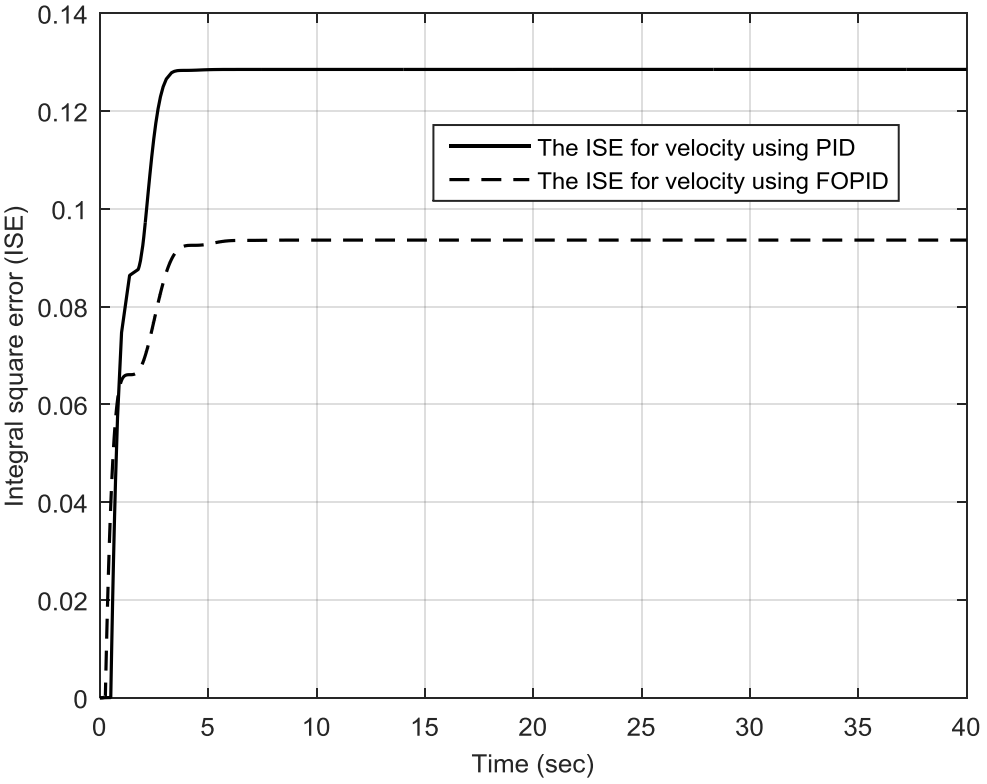



Fig. 4.22 ISE of the velocity for linear trajectory PID and FOPID controllers.

### *4.7.2 Application Case 2*

A comparison is conducted between conventional and fractional order PID controller for a circular smooth trajectory as depicted in Fig. 4.23. This trajectory is performed a continuous gradient trajectory motion. Hence, the error between the actual and desired trajectory is supplied into the first fractional order PID controller. The output of the fractional order controller is presented the control action that is connected the actuator of the wheel. Fig. 4.24 demonstrates the orientation of an unmanned ground vehicle that it tracks and converges to the desired circular trajectory. The error between the desired and actual orientations for both the PID and fractional order PID controllers is depicted in Fig. 4.25. It can be observed that the orientation errors are maintained around zero.

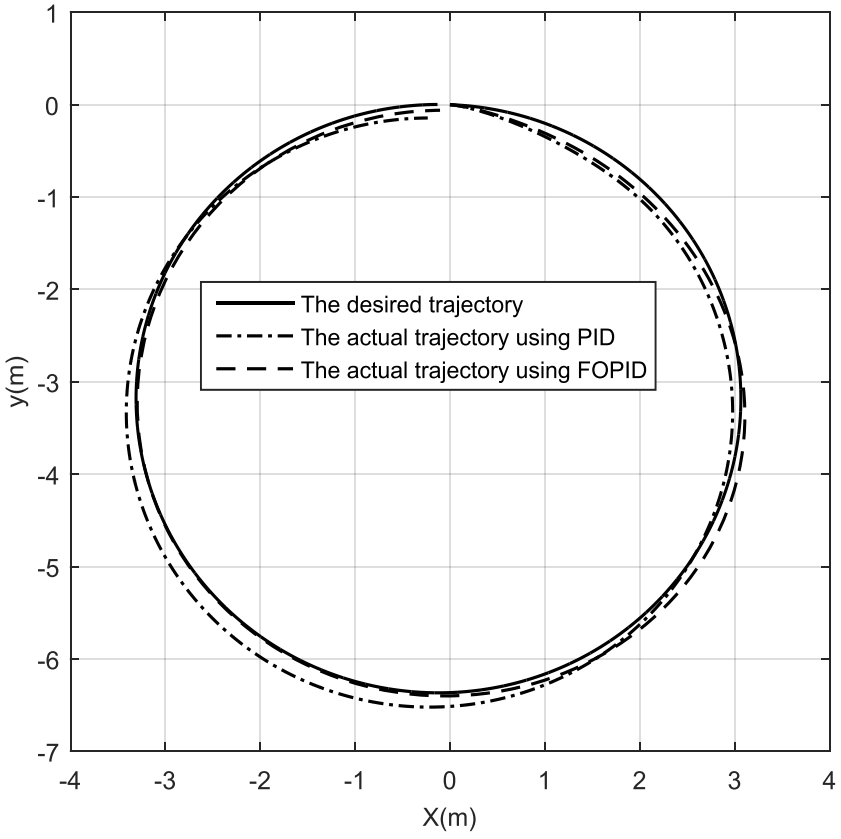



Fig. 4.23 Circular trajectories using PID and FOPID controllers.

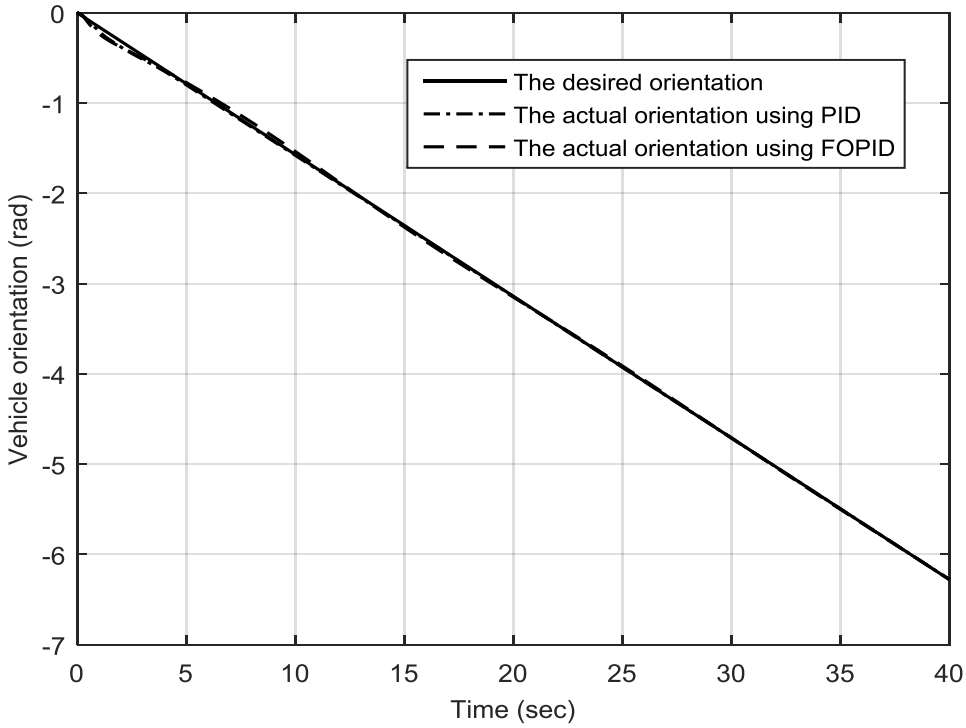



Fig. 4.24 Orientations for circular trajectory using PID and FOPID controllers.

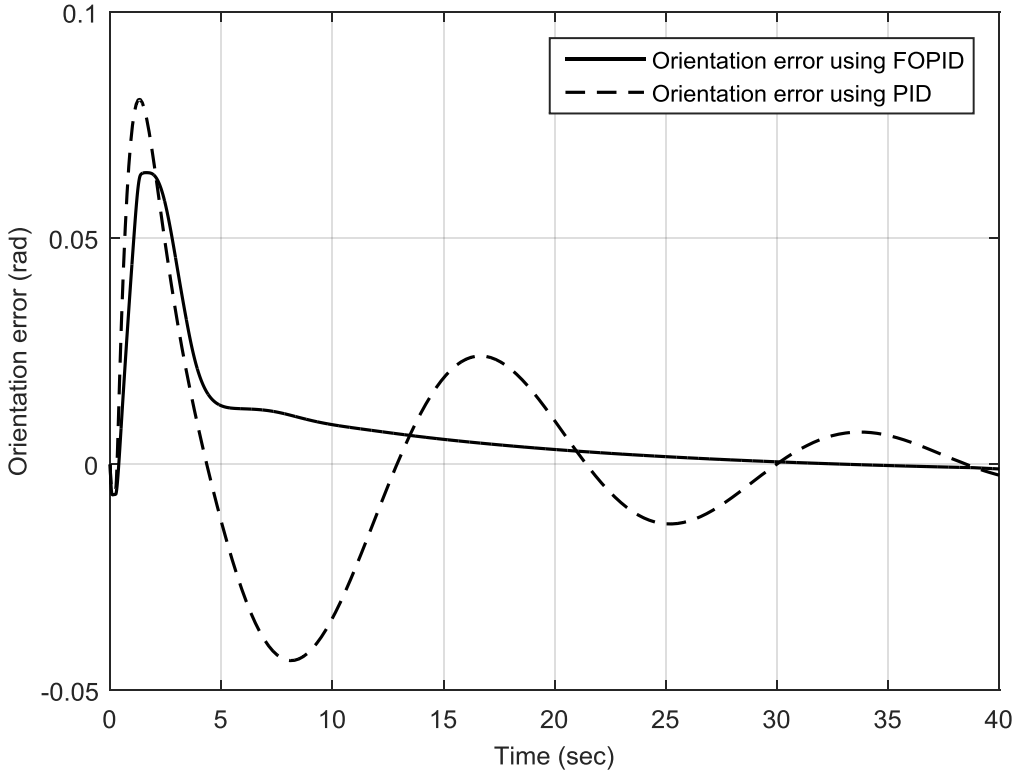



Fig. 4.25 Error in the orientation for circular trajectory using PID and FOPID controllers.

Control efforts for the left and right wheels for the circular trajectory based on using conventional and fractional order PID controllers are shown in Fig. 4.26 and Fig. 4.27, respectively. There are no significant differences between the control actions of the two control methodologies.

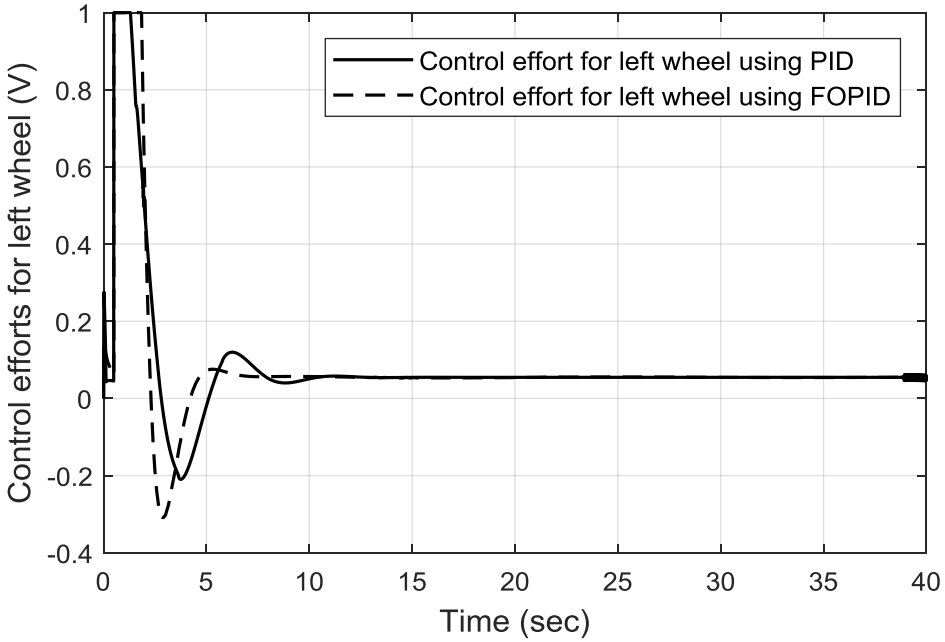



Fig. 4.26 Control efforts for left wheel of the circular trajectory using PID and FOPID controllers.

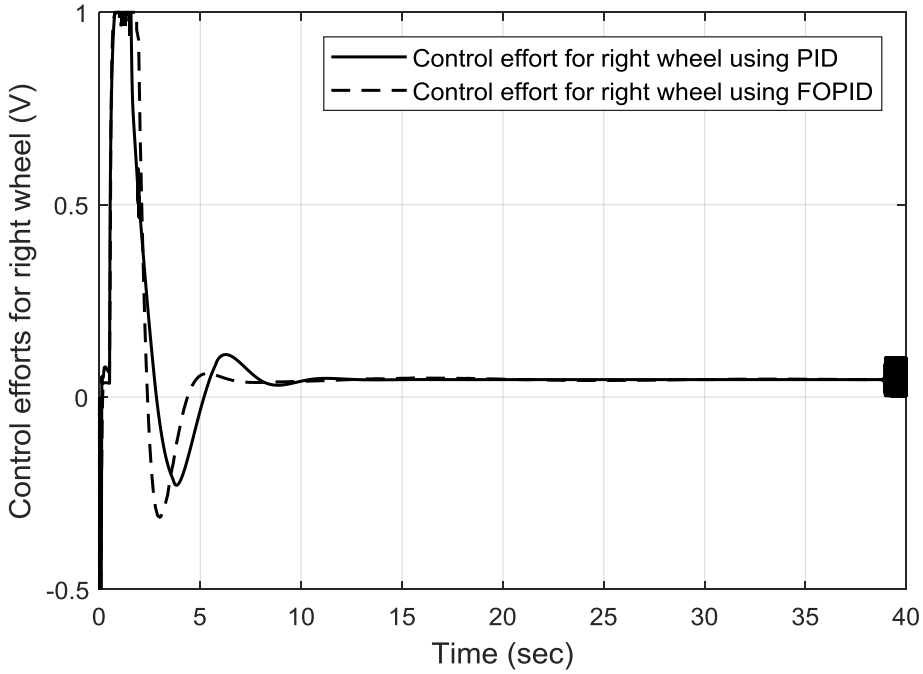



Fig. 4.27 Control efforts for right wheel of the circular trajectory using PID and FOPID controllers.

The second input of the system represents the desired velocity and it equals to $v_d = 0.5$ [m/sec] for interval $0 \leq t \leq 40$ $[sec]$. The velocity is compared with the actual velocity based on the traditional and the fractional order PID controller as illustrated in Fig. 4.28. The produced error of comparing the desired and actual velocities shown in Fig. 4.29 is supplied to the second fractional order controller. The actual velocities of both controllers are retrieved from its maximum amplitude and approached zero value after '6 sec'. Similarly, a comparison of trajectory tracking is conducted for both of X and Y coordinates as shown in Figs. 4.30 and 4.31. Hence, the trajectory tracking errors can be determined and are demonstrated in Figs. 4.32 and 4.33. Finally, the comparison of the changing of the integral square error functions of both the orientation and velocity is depicted in Figs. 4.34 and 4.35.

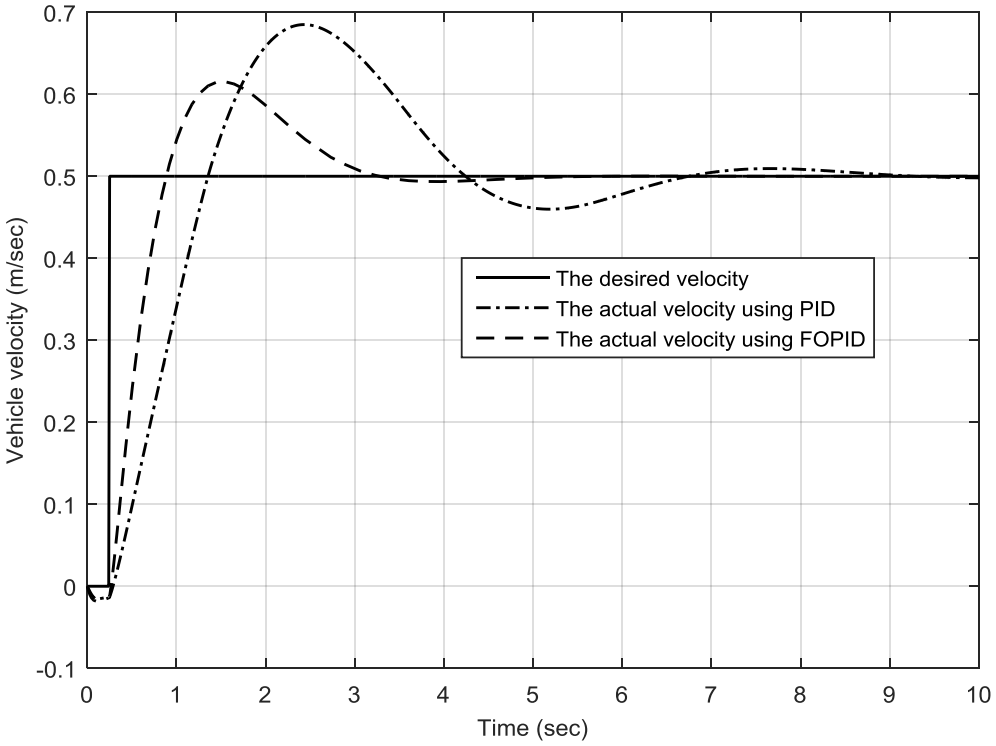



Fig. 4.28 Velocities for circular trajectory using PID and FOPID controllers.

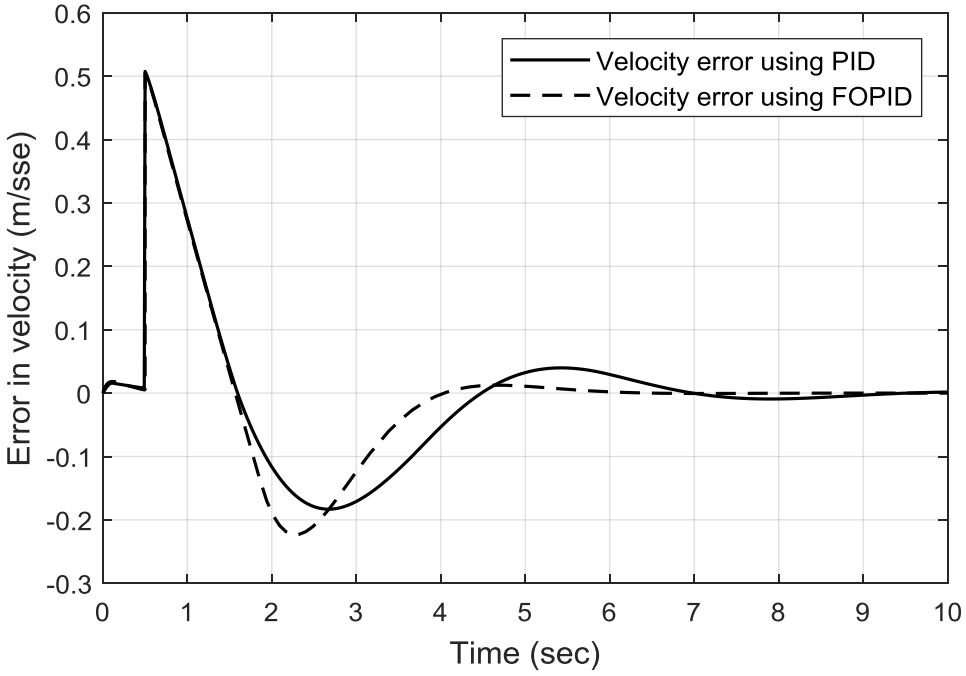



Fig. 4.29 Error in velocity for circular trajectory using PID and FOPID controllers.

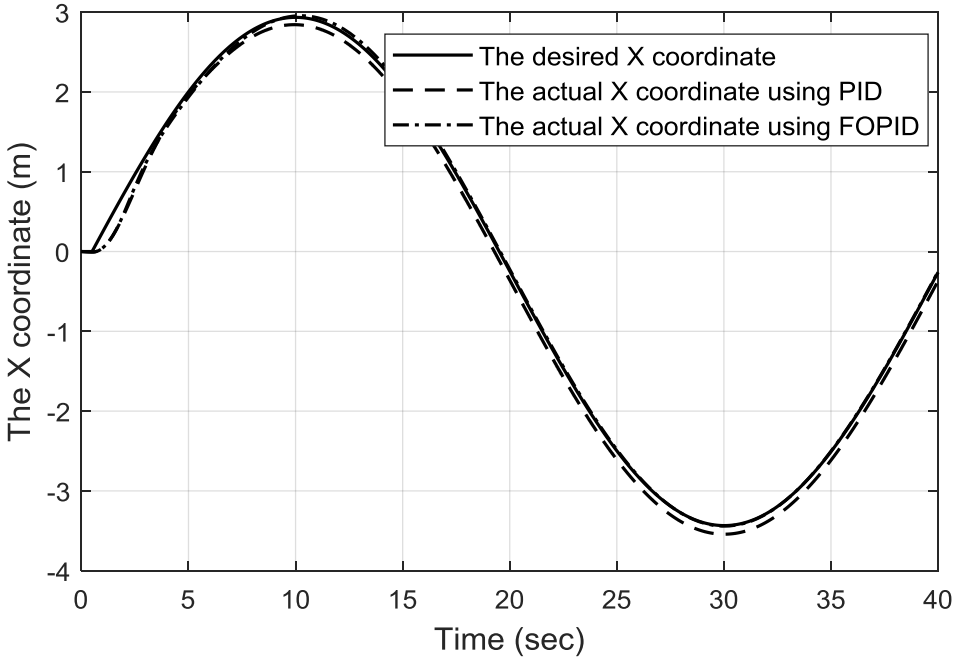



Fig. 4.30 X-coordinates for circular trajectory using PID and FOPID controllers.

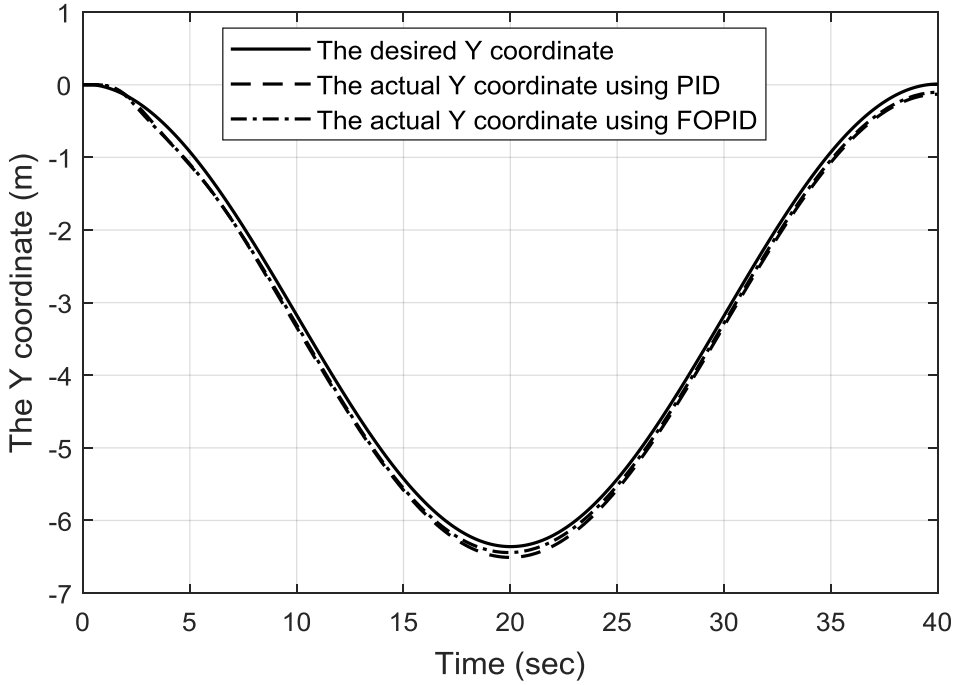



Fig. 4.31 Y-coordinates for circular trajectory using PID and FOPID controllers.

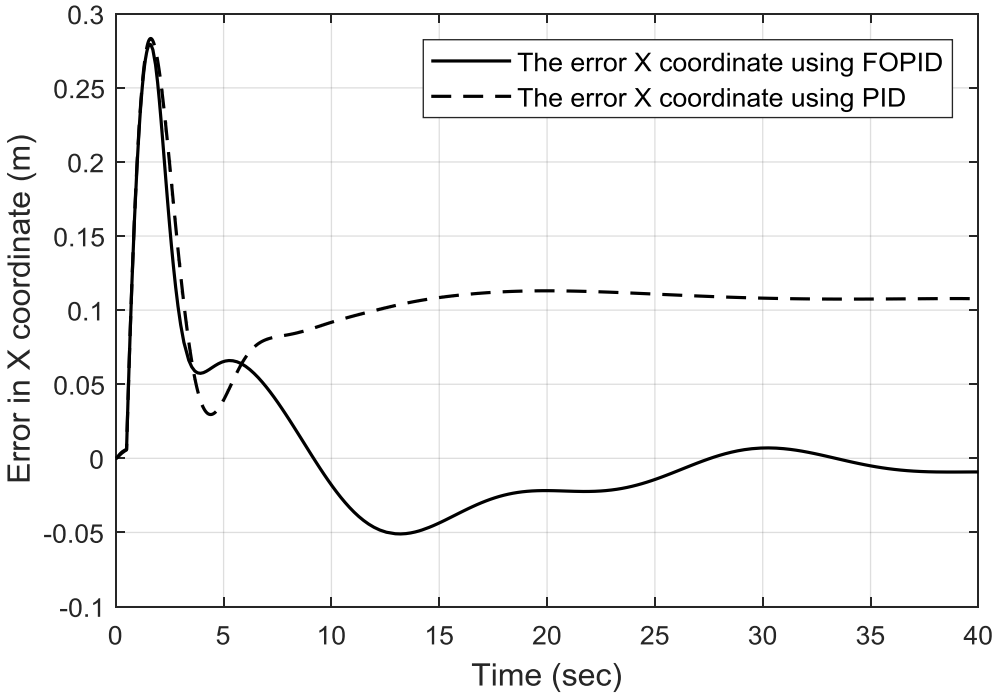



Fig. 4.32 Error in X-coordinates for circular trajectory using PID and FOPID controllers.

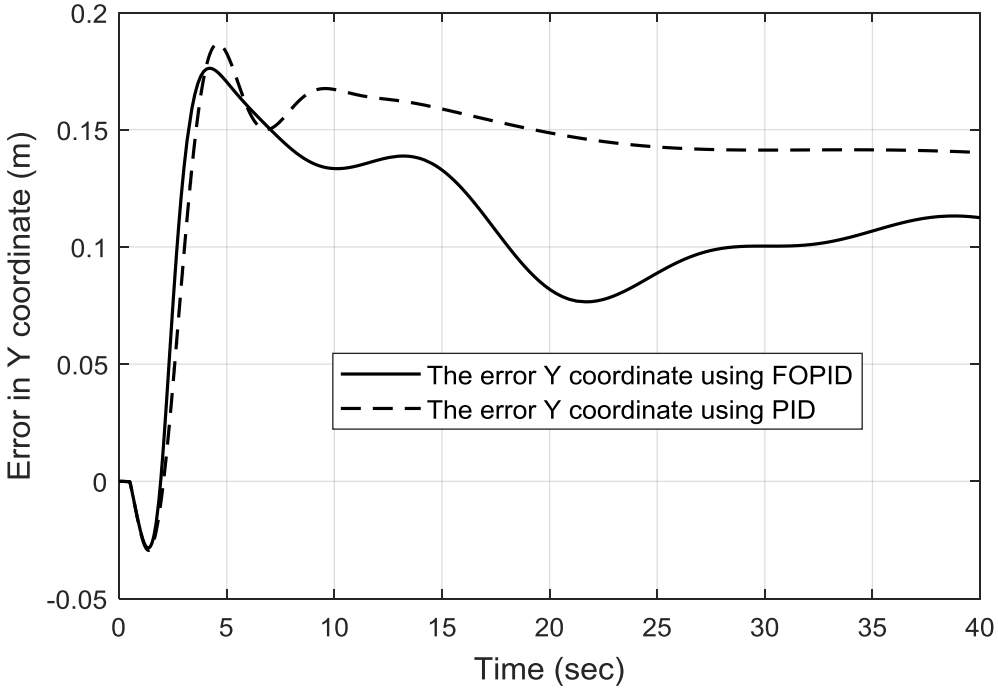



Fig. 4.33 Error in Y-coordinates for the circular trajectory using PID and FOPID controllers.

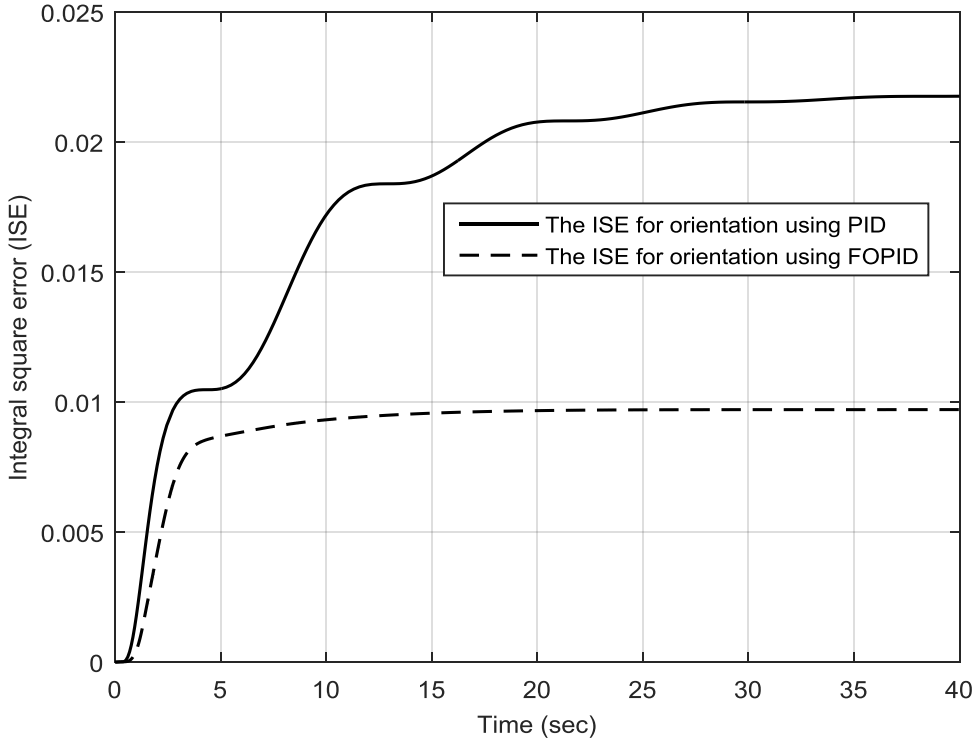



Fig. 4.34 ISE of orientation for circular trajectory using PID and FOPID controllers.

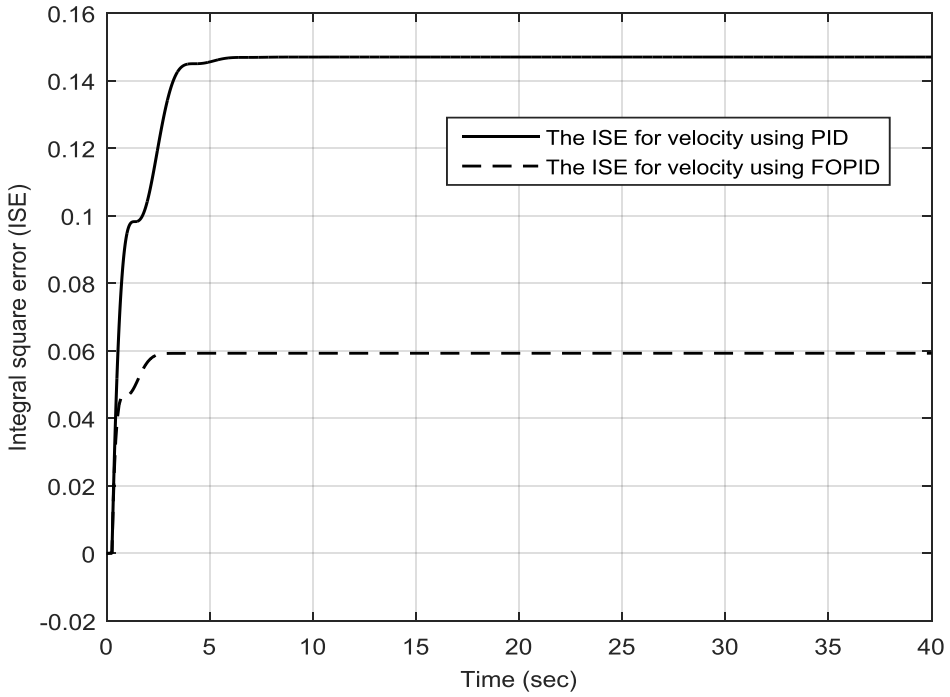



Fig. 4.35 ISE of velocity for circular trajectory using PID and FOPID controllers.

### *4.7.3 Application Case 3*

In this case, a lemniscate trajectory is utilised as a new reference input. It behaves like the circular trajectory in terms of a continuous gradient motion. However, it demonstrates varied rotations to complete its cycle of movement. Fig. 4.36 illustrates the performance of the traditional and fractional order PID controllers of guiding the motion of a UGV. In fact, both have shown a precise performance in terms of trajectory tracking and minimising the tracking error. However, the fractional order PID controller displays better results based on the orientation and its error that are shown in Fig. 4.37 and Fig. 4.38, respectively. Control efforts for the left and right wheels for the lemniscate trajectory based on using conventional and fractional order PID controllers are shown in Fig. 4.39 and Fig. 4.40, respectively. It can be clearly that the control actions have been improved by using FOPID. The overshoot of both graphs is minimised and hence, better time responses are obtained.

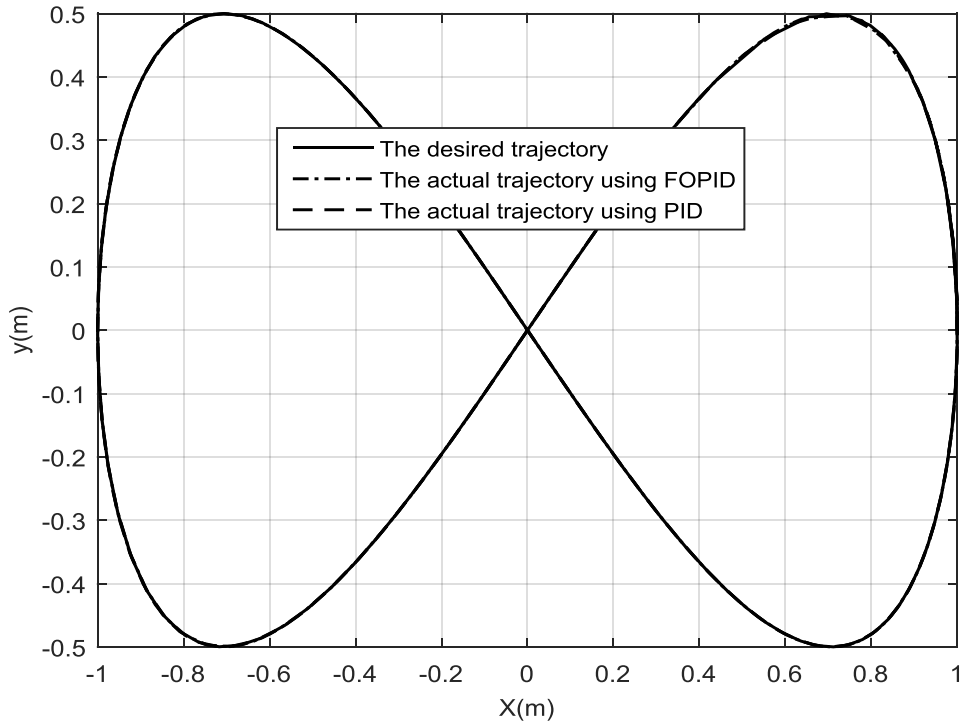



Fig. 4.36 Lemniscate trajectories PID and FOPID controllers.

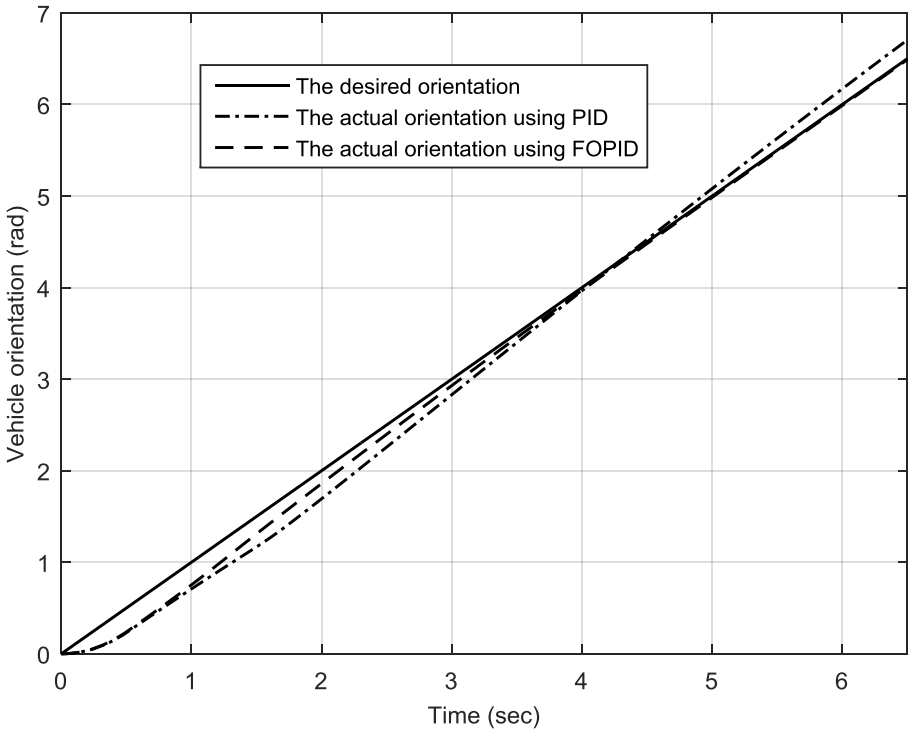



Fig. 4.37 Orientations for the lemniscate trajectory using PID and FOPID controllers.

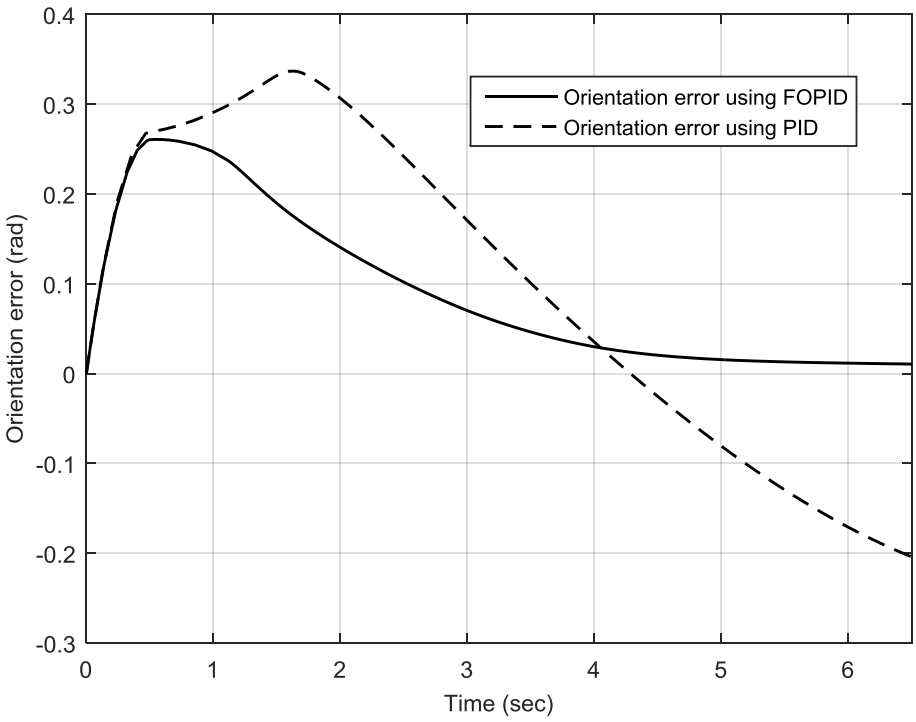



Fig. 4.38 Error in orientations for the lemniscate trajectory using PID and FOPID controllers.

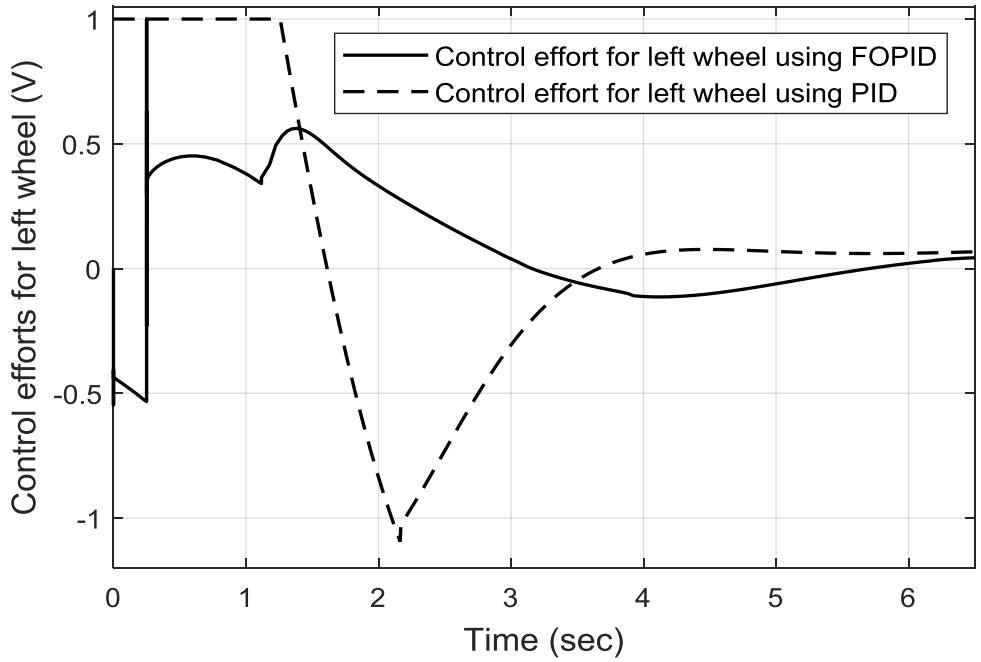



Fig. 4.39 Control efforts for left wheel of lemniscate trajectory using PID and FOPID controllers.

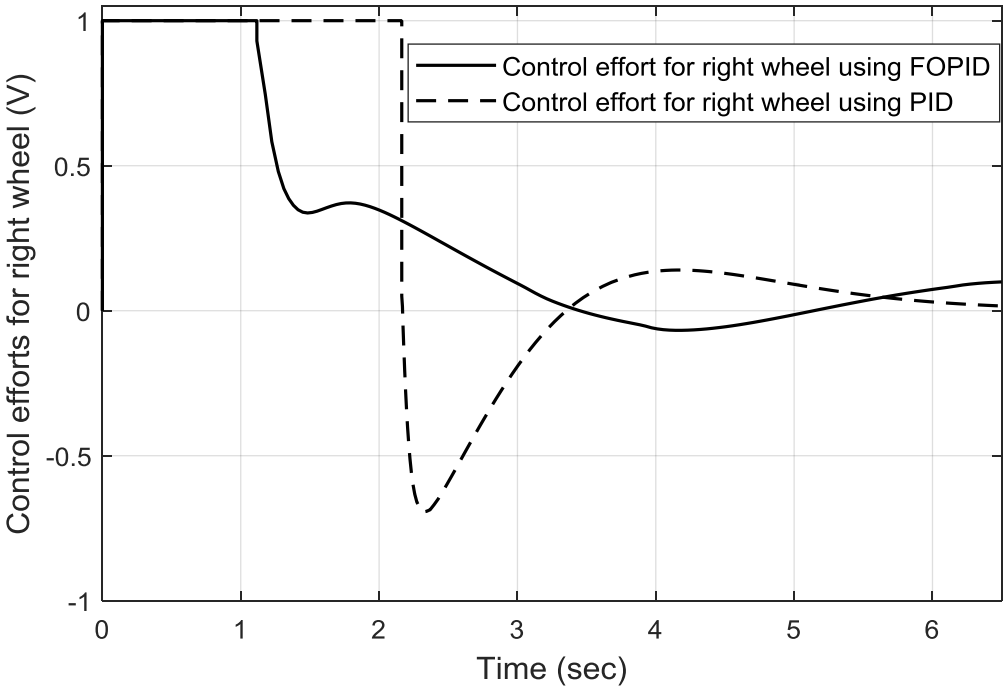



Fig. 4.40 Control efforts for right wheel of lemniscate trajectory using PID and FOPID controllers.

Likewise, the desired and the actual velocities are compared using the same approach as demonstrated in Fig. 4.41. Besides, the error between the desired and the actual velocities proves that the fractional order methodology performs a more effective response as illustrated in Fig. 4.42. Correspondingly, the trajectory tracking of X and Y coordinates error is provided from Fig. 4.43 to Fig. 4.46. Comparably, the obtained error for each coordinate confirms that the fractional order has improved the performance of the system by minimising the tracking error. Eventually, the changing of the integral square error functions for both the orientation and velocity are given in Figs. 4.47 and 4.48. It is noticeable that the fractional order PID controller has introduced advantageous of minimising expecting errors.

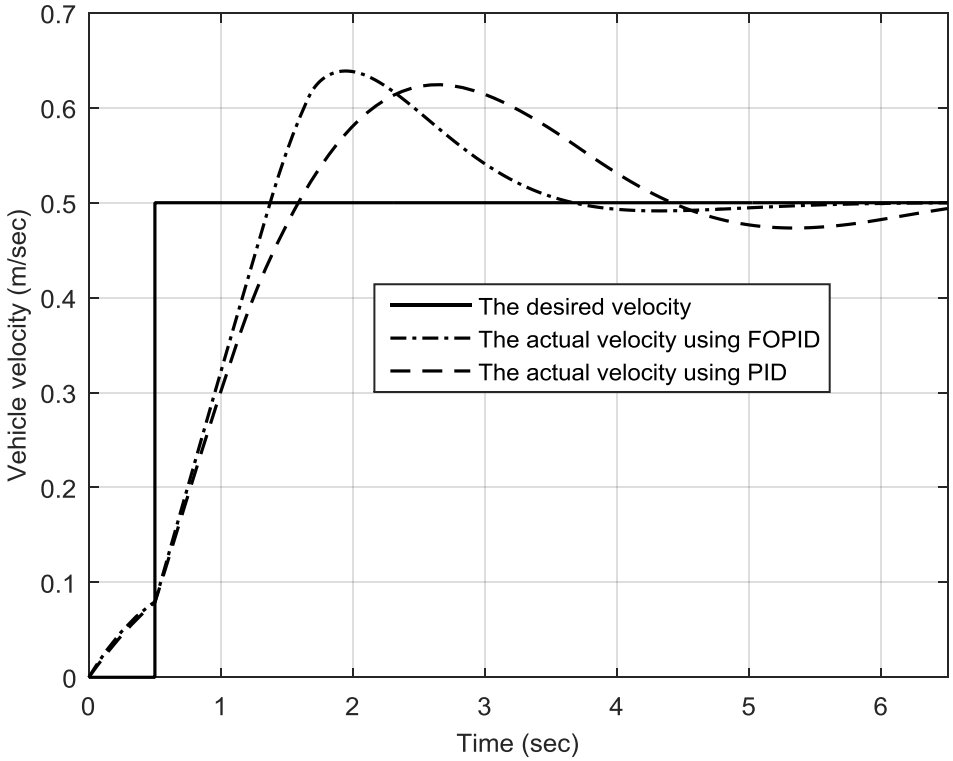



Fig. 4.41 Velocities for the lemniscate trajectory using PID and FOPID controllers.

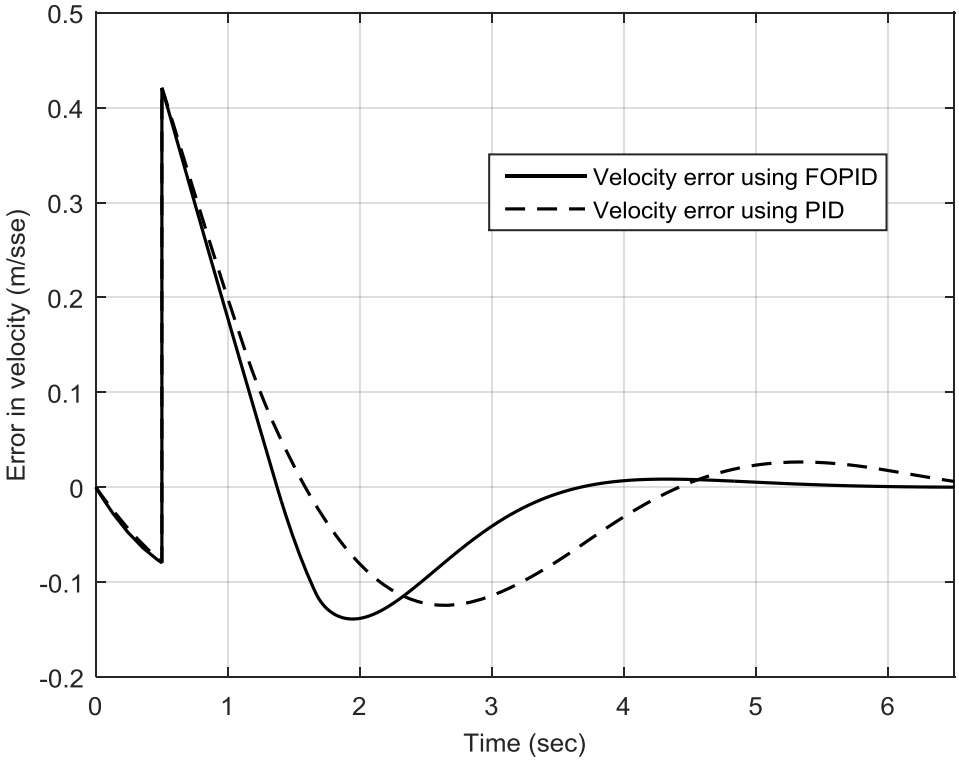



Fig. 4.42 Error in velocities for the lemniscate trajectory using PID and FOPID controllers.

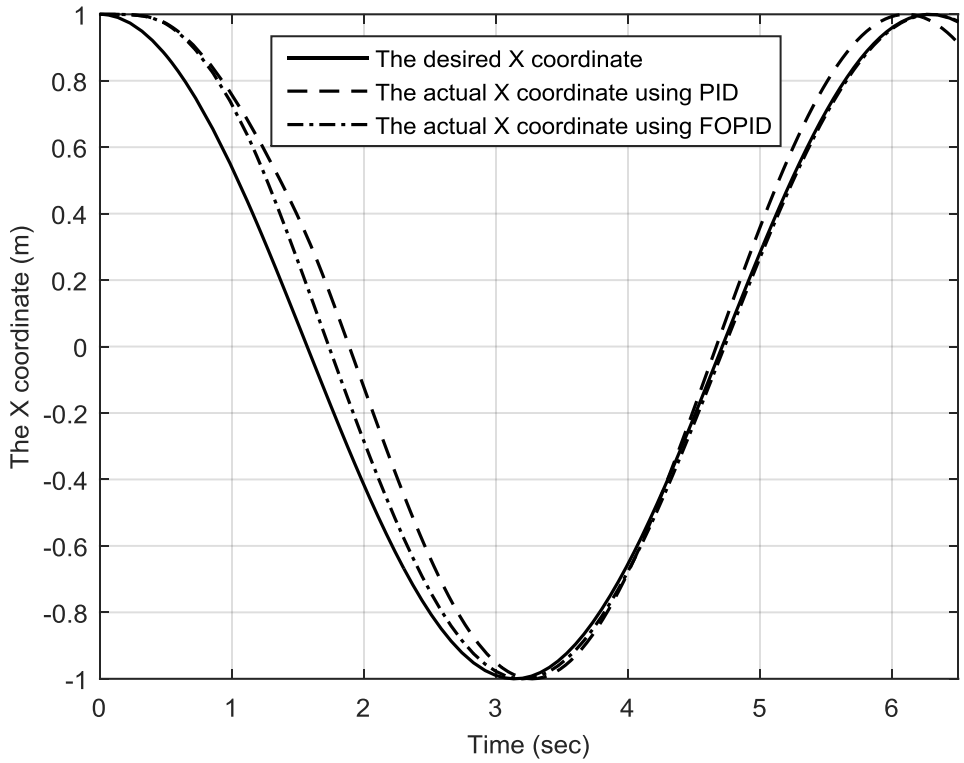



Fig. 4.43 X-coordinates for the lemniscate trajectory using PID and FOPID controllers.

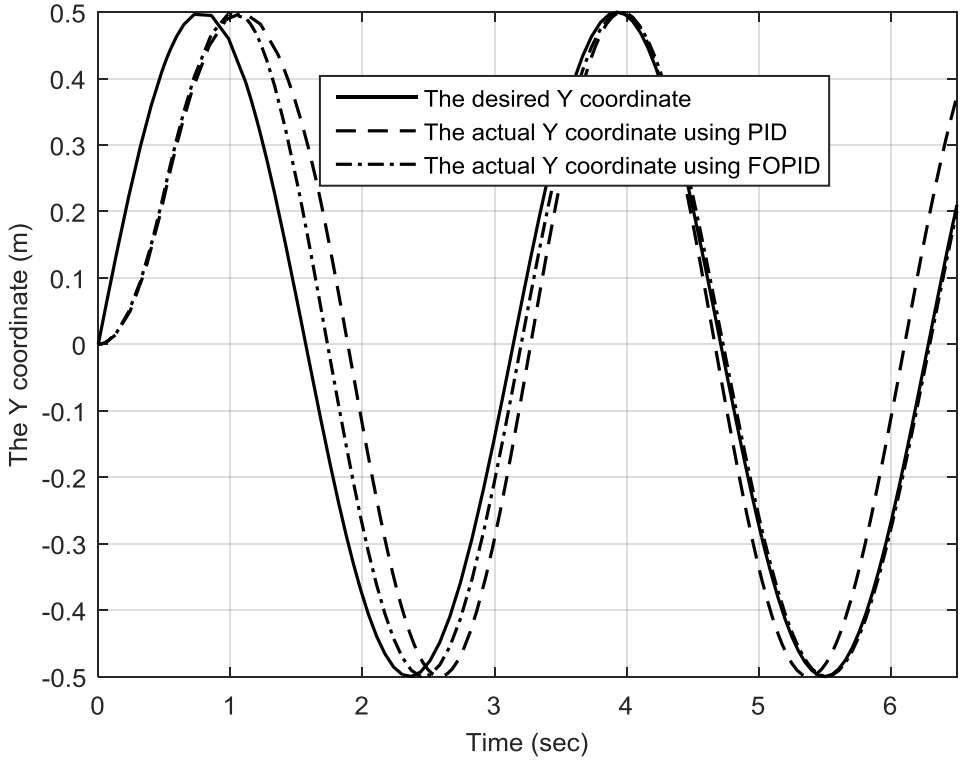



Fig. 4.44 Y-coordinates for the lemniscate trajectory using PID and FOPID controllers.

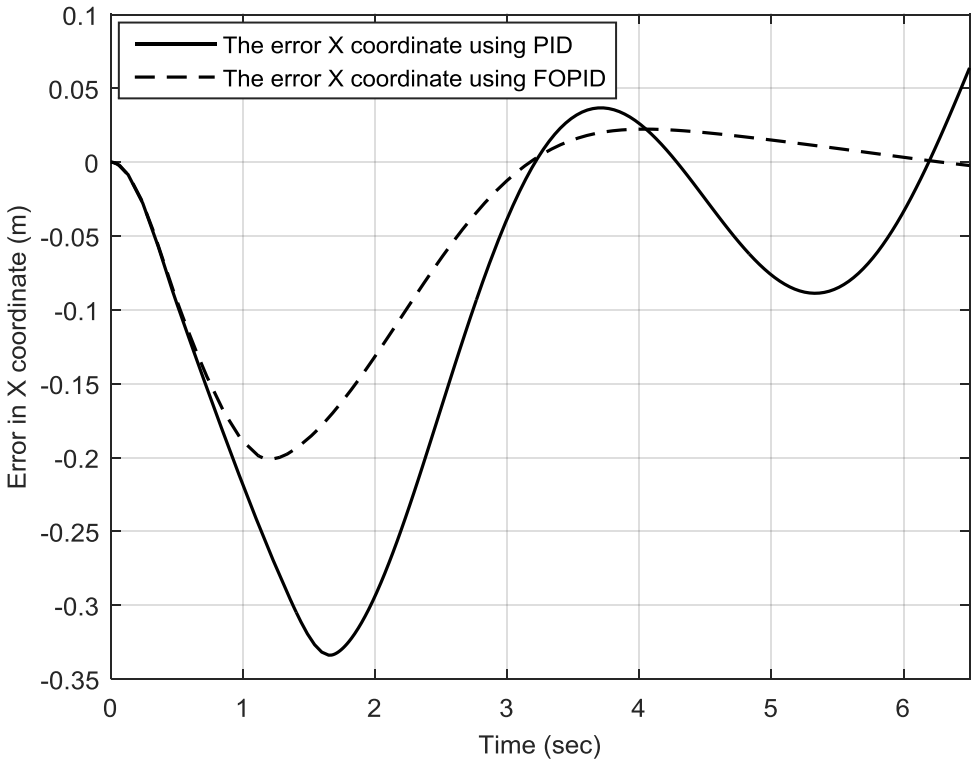



Fig. 4.45 Error of X-coordinates for lemniscate trajectory using PID and FOPID controllers.

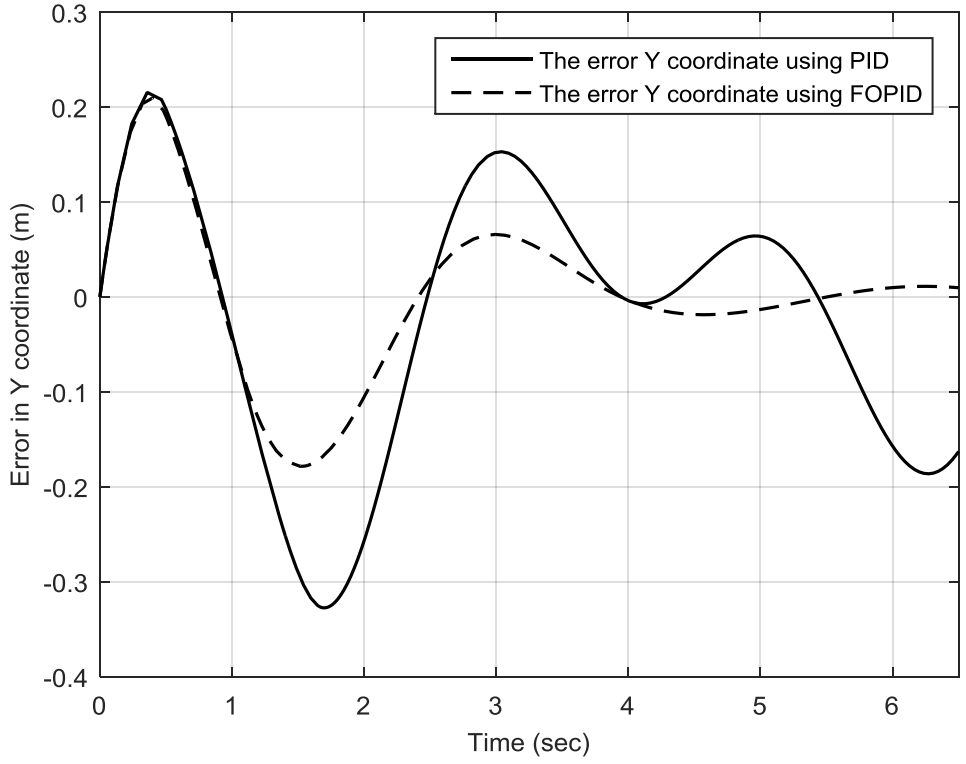



Fig. 4.46 Error of Y-coordinates for lemniscate trajectory using PID and FOPID controllers.

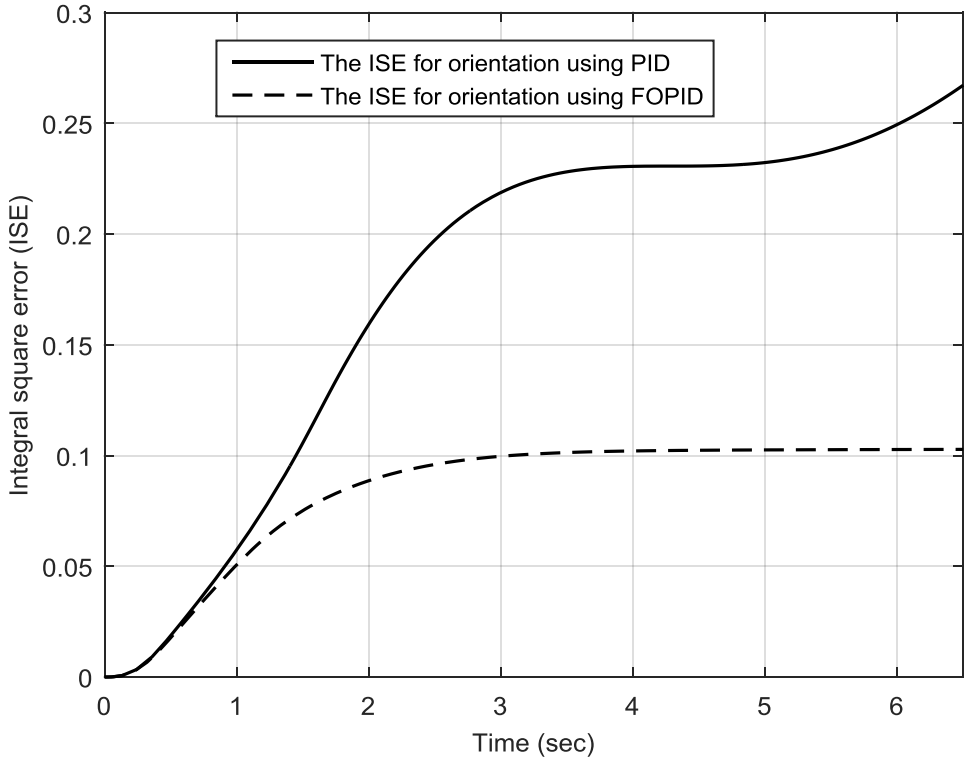



Fig. 4.47 ISE of the orientation for lemniscate trajectory using PID and FOPID controllers.

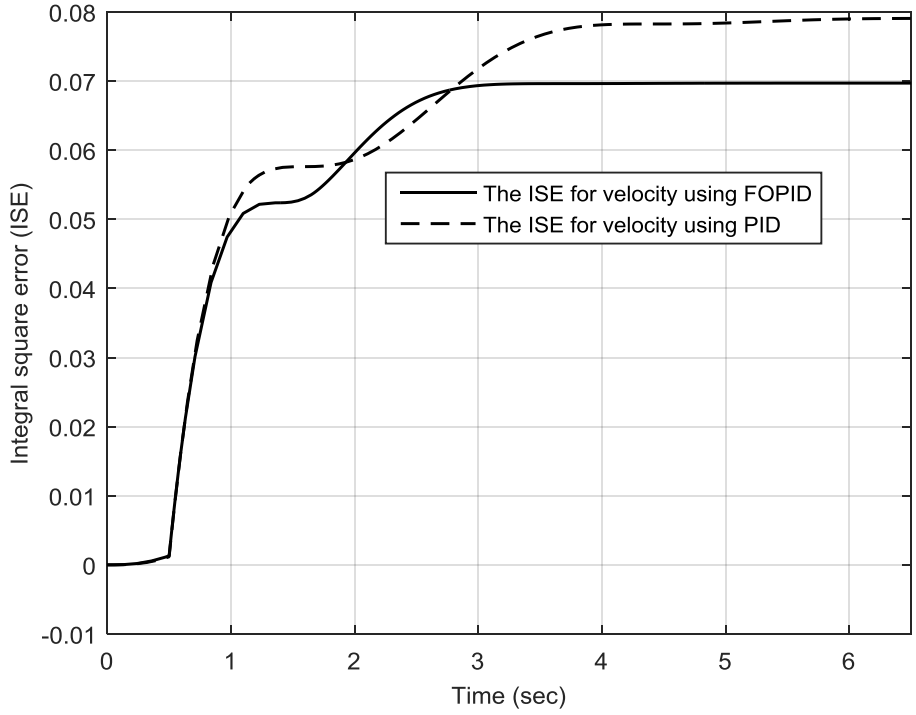



Fig. 4.48 ISE of velocity for lemniscate trajectory using PID and FOPID controllers.

### 4.7.4 Application Case 4

In this case, a reference square trajectory is considered. Obviously, it is an explicit example of a non-continuous gradient trajectory. The main complexity of such a trajectory results from a sharp and a non-continuous motion. In Chapter 3, it has been noticed that the trajectory tracking of the square trajectory led to a significant error using the traditional PID controller. Based on the fractional order PID controller, the trajectory tracking errors and the control efforts have been improved significantly. Nonetheless, the trajectory tracking still suffers from high values of tracking error that might produce an unstable and a disturbed motion. Fig. 4.49 demonstrates the difference between the actual velocities of the conventional and fractional order PID controllers comparing to the desired square trajectory.

In addition, the introduced graph for the orientation as shown in Fig. 4.50 demonstrates similar facts about the performance of the UGV and its function in response to the non-continuous gradient trajectory. The tracking error response that is depicted in Fig. 4.51 validates the effectiveness of the fractional order PID controller and its ability to minimise the tracking error significantly. The value of the trajectory tracking error reaches the peak at each side of the desired square trajectory whilst it attempts to track the trajectory. Actually, an observable reason that describes this phenomenon, it belongs to the sudden changing at the corners of the desired trajectory. Hence, the UGV has to accomplish an appropriate and a feasible turn to retrieve its normal posture. Along any one side of the square, the desired orientation angle is constant, therefore, the orientation error approaches zero. However, at the end of one side of the square trajectory, the desired orientation angle changes suddenly. Therefore, the orientation error of the UGV against the desired trajectory at the corners of the square is increased.

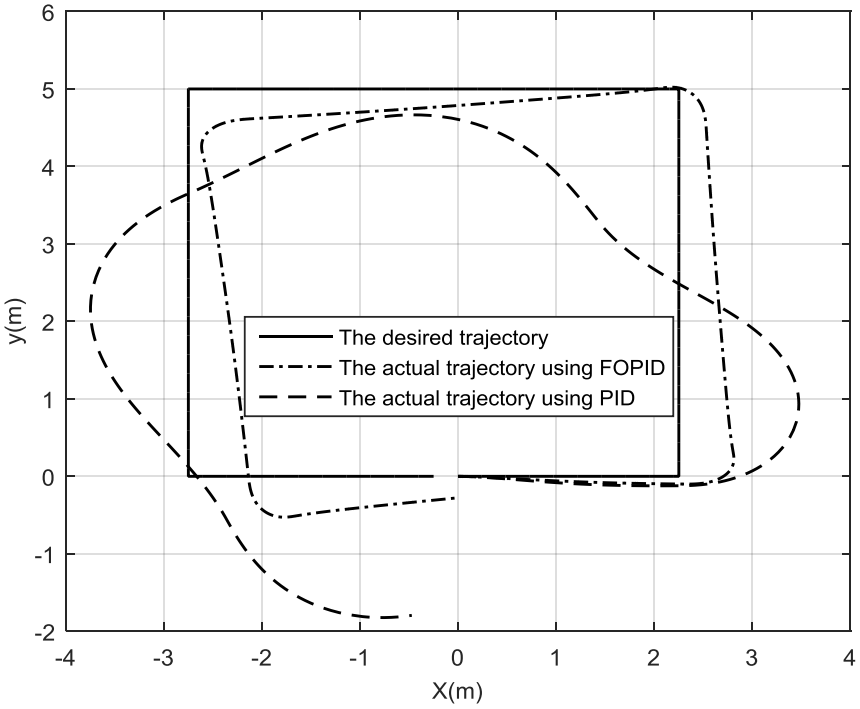



Fig. 4.49 Square trajectories using PID and FOPID controllers.

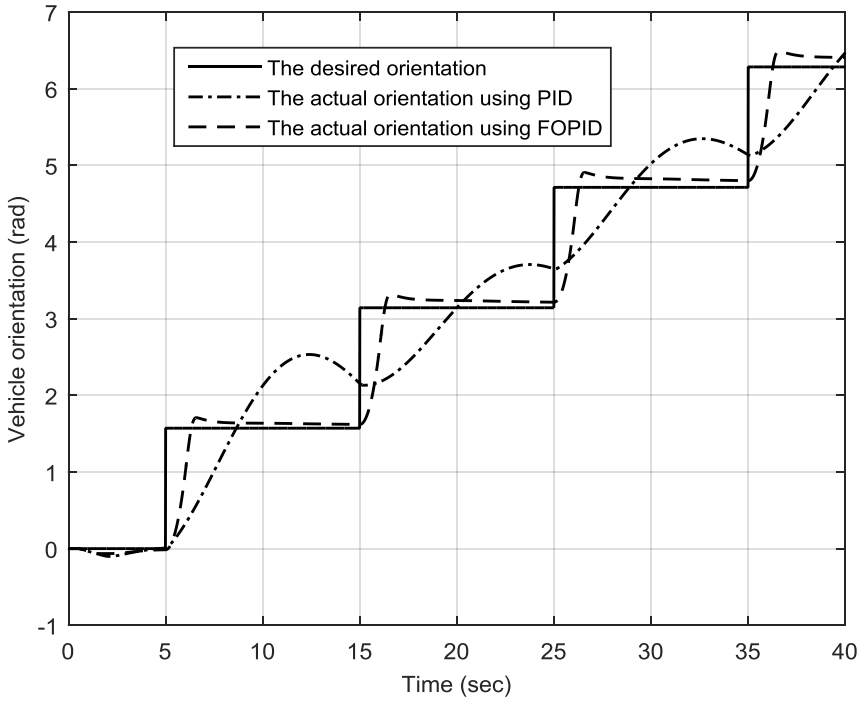



Fig. 4.50 Orientations for square trajectory using PID and FOPID controllers.

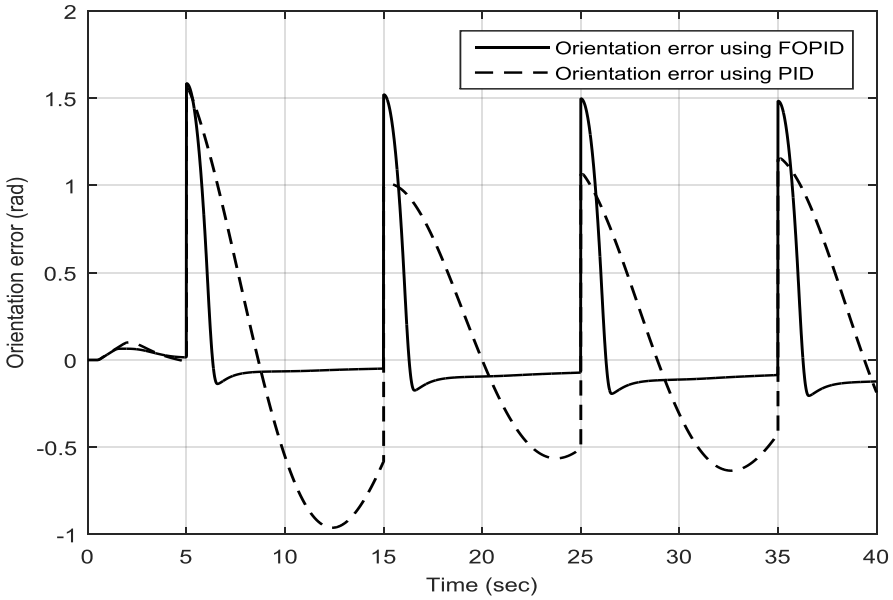



Fig. 4.51 Error in orientations for the square trajectory using PID and FOPID controllers.

Control efforts for the left and right wheels for the square trajectory are demonstrated in Fig. 4.52 and Fig. 4.53, respectively. It can be clearly that the control actions have been improved by using FOPID controller comparing to conventional PID controller. The simulation results have demonstrated the effectiveness of the proposed controller by showing its ability to generate small values of the control input torques for right and left wheels with small sharp spikes at the corners of changing the gradient of the non-continuous square trajectory.

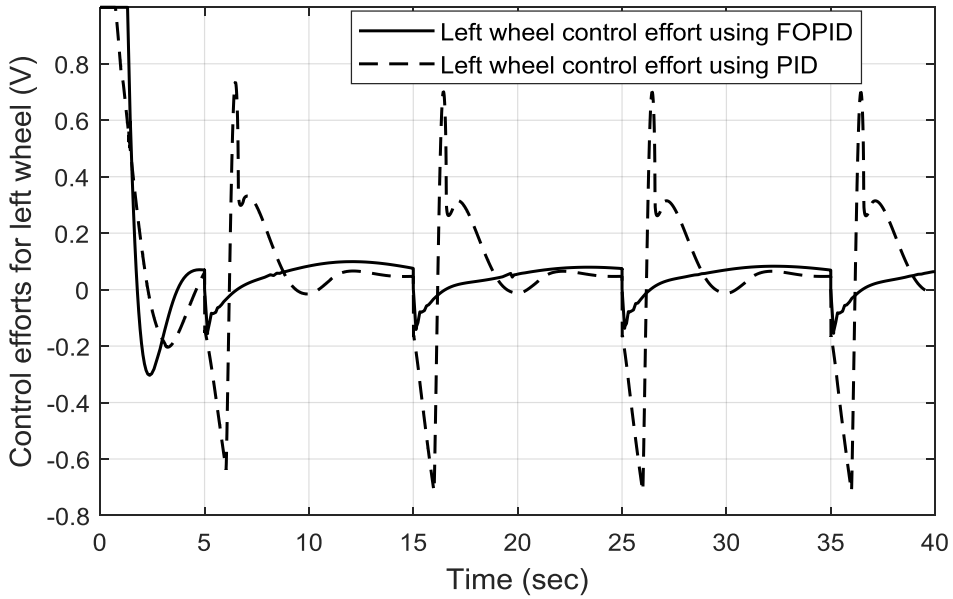



Fig. 4.52 Control efforts for left wheel of square trajectory using PID and FOPID controllers.

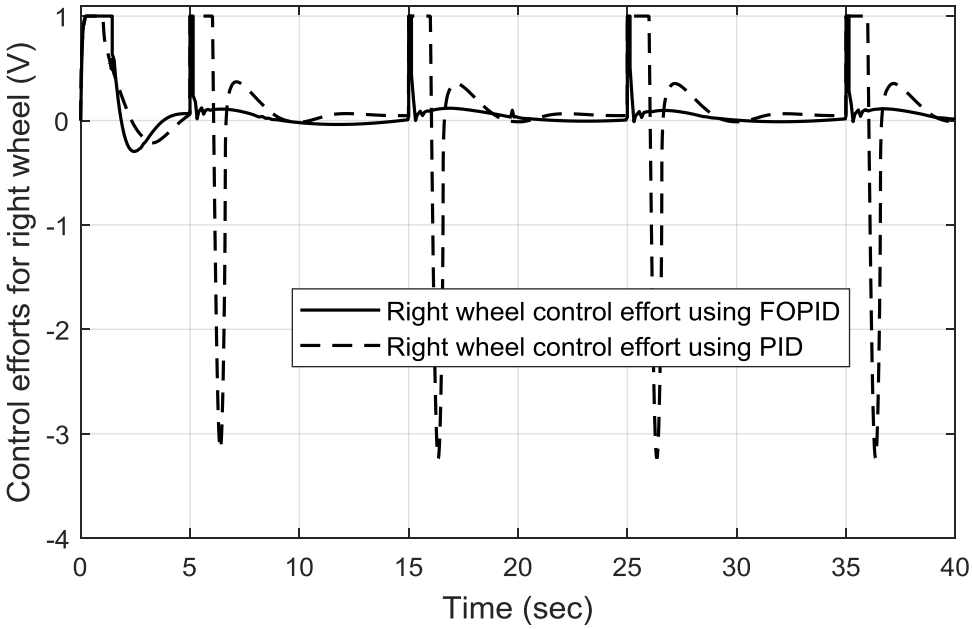



Fig. 4.53 Control efforts for right wheel of square trajectory using PID and FOPID controllers.

The velocities of the UGV based on the classic and the fractional order PID controller are shown in Fig. 4.54. Distinctly, the error of velocities for both of control methodologies validate that the fractional order PID controller has made a noticeable improvement to track the desired velocity as depicted in Fig. 4.55. After a duration of '5 sec', the peak value of the velocity error has a range between '-0.12 m/sec' to '0.18 m/sec' using the traditional PID controller. Whereas, it has a value of '-0.02 m/sec' using the fractional order PID controller.

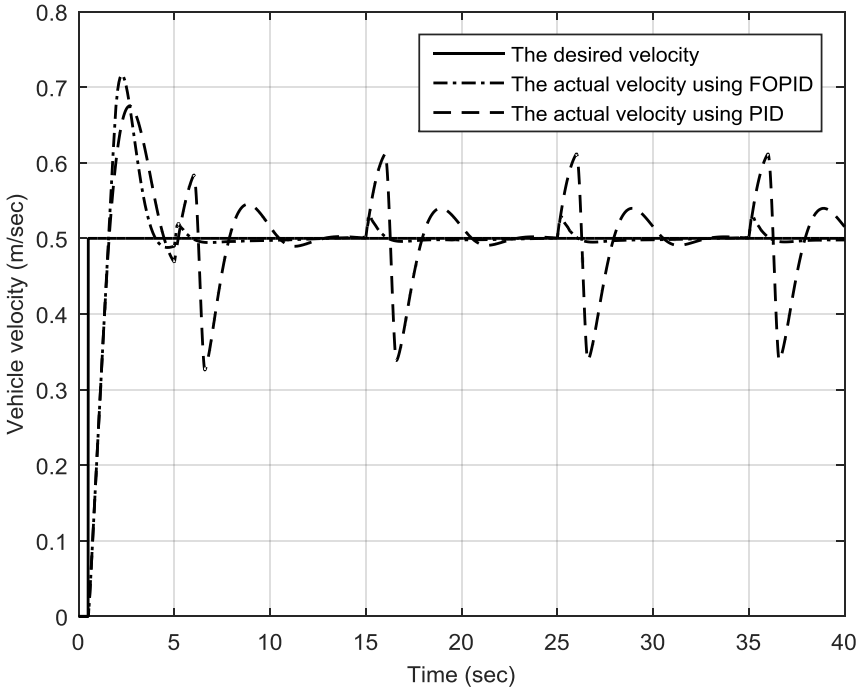



Fig. 4.54 Velocities for square trajectory using PID and FOPID controllers.

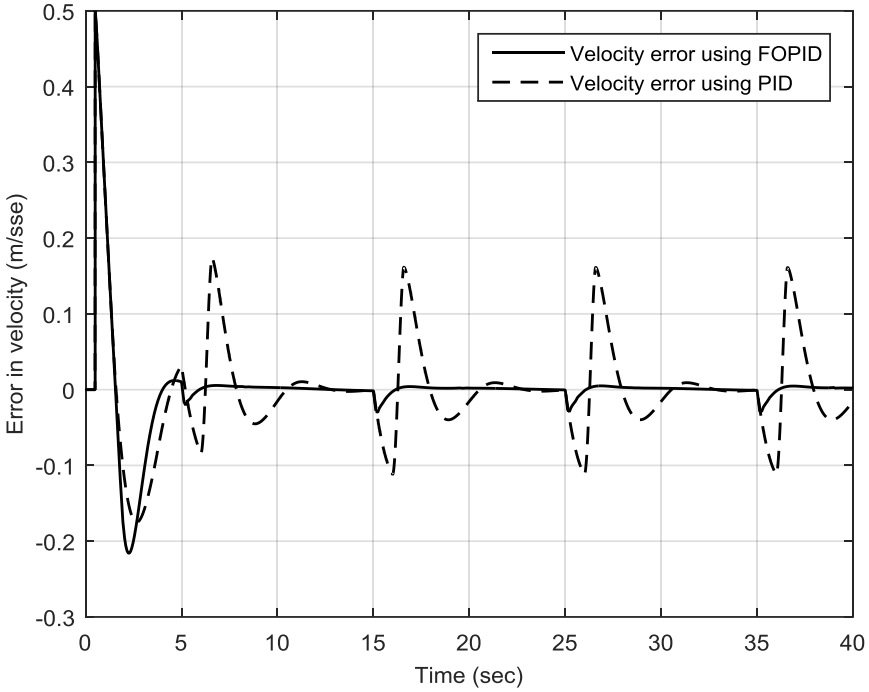



Fig. 4.55 Error in velocity for square trajectory using PID and FOPID controllers.

Comparisons of trajectory tracking for X and Y coordinates are carried out as illustrated in Fig. 4.56 and Fig. 4.57, respectively. The tracking errors have been reduced to small values as shown in Fig. 4.58 and Fig. 4.59, for position tracking errors. The value of X-coordinate error in the square trajectory is oscillated around '-0.5m' using the fractional order PID controller whilst it drops to '-1m' and reaches a maximum of '1m' using the conventional PID controller. The Y-coordinate error in the square trajectory is also fluctuated but around '0.5m' using the fractional order PID controller. However, it exceeds a value of '1.5m' based on the conventional PID controller. The final comparisons are conducted for the integral square error of the orientation and velocity. They have approved the validation of our proposed methodology as demonstrated in Fig. 4.60 and Fig. 4.61, respectively. It is noteworthy that at the corner sides of the desired square, the tracking error rates have been improved and enhanced.

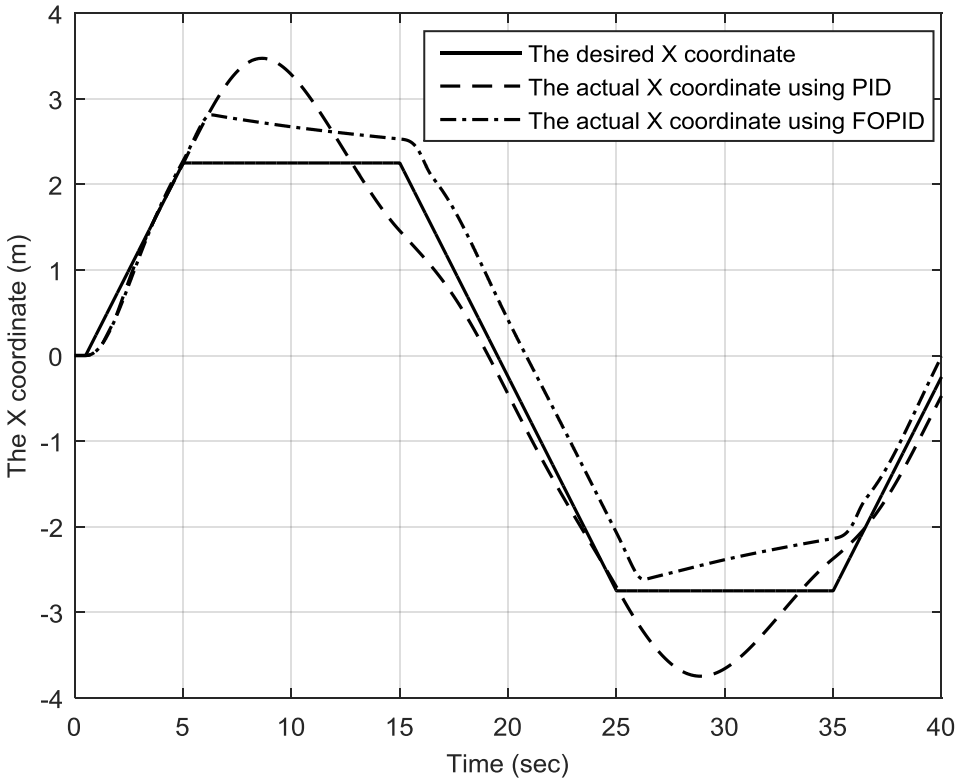



Fig. 4.56 X-coordinates for square trajectory using PID and FOPID controllers.

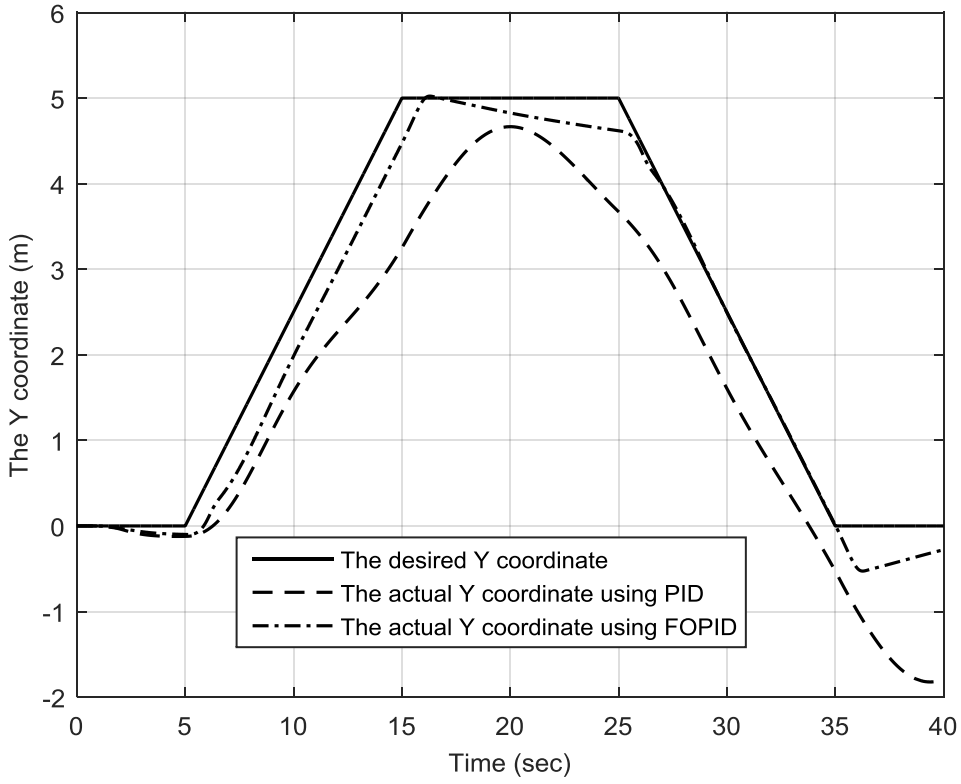



Fig. 4.57 Y-coordinates for square trajectory using PID and FOPID controllers.

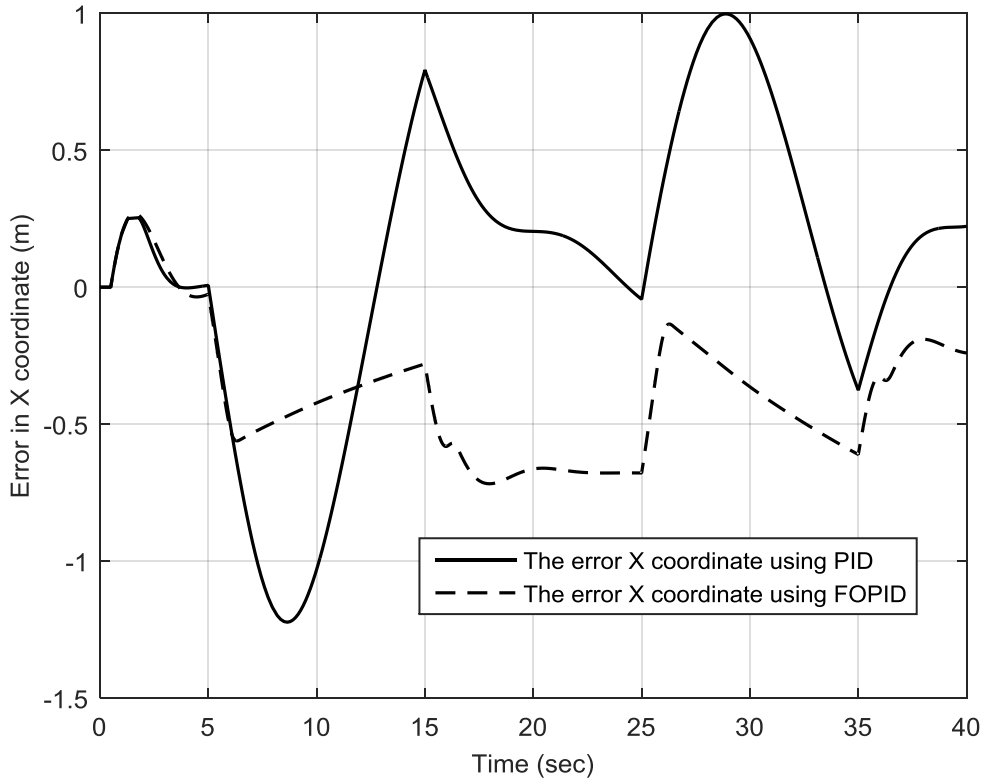



Fig. 4.58 Error in X coordinate for the square trajectory using PID and FOPID controllers.

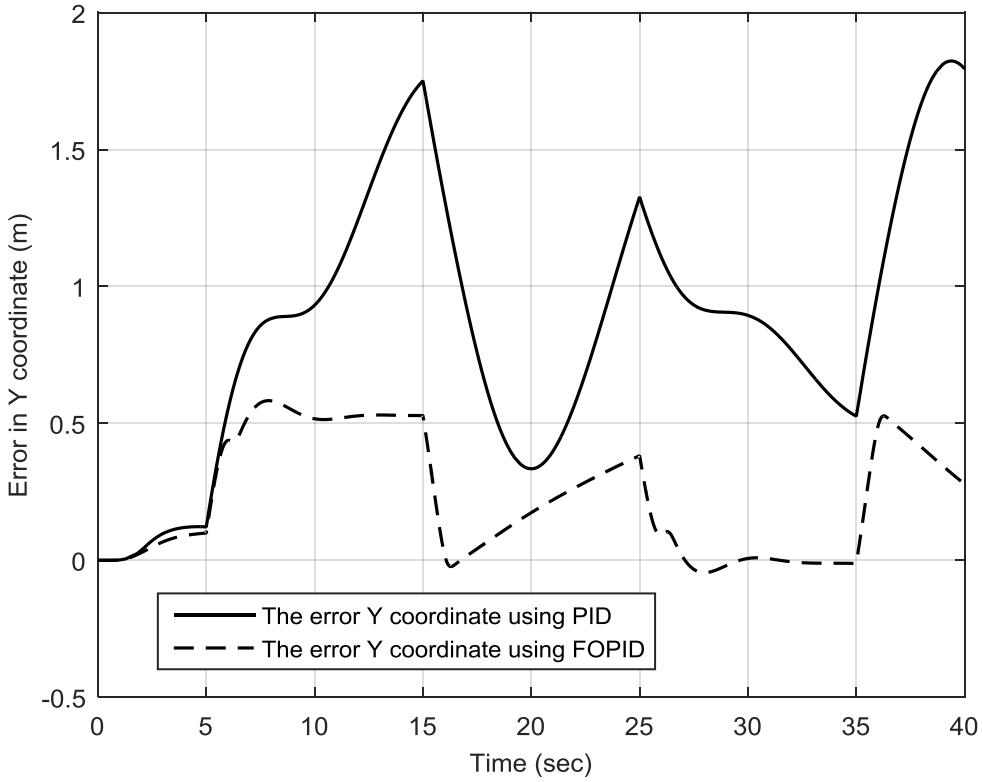



Fig. 4.59 Error in Y coordinates for the square trajectory using PID and FOPID controllers.

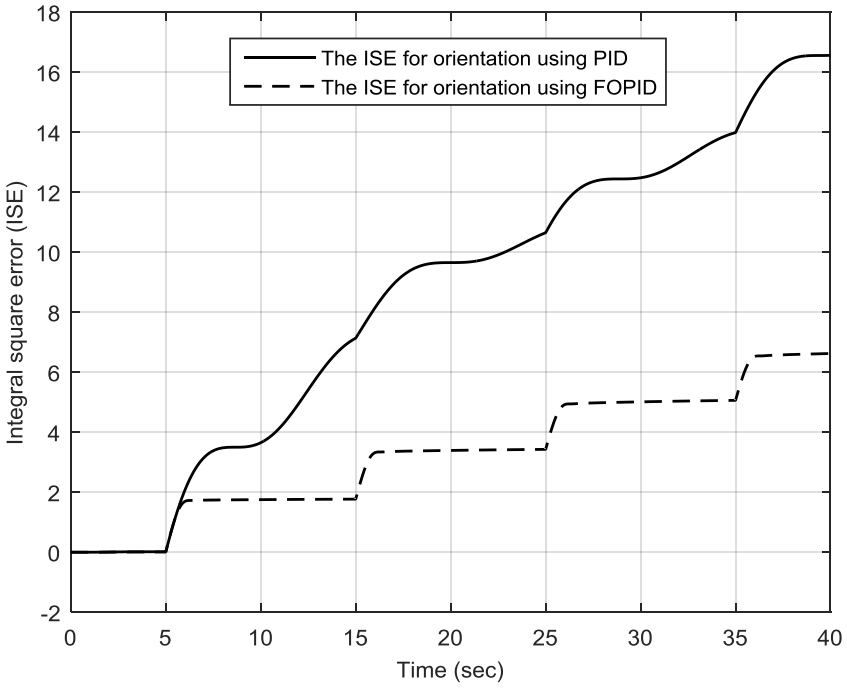



Fig. 4.60 ISE of orientation for square trajectory using PID and FOPID controllers.

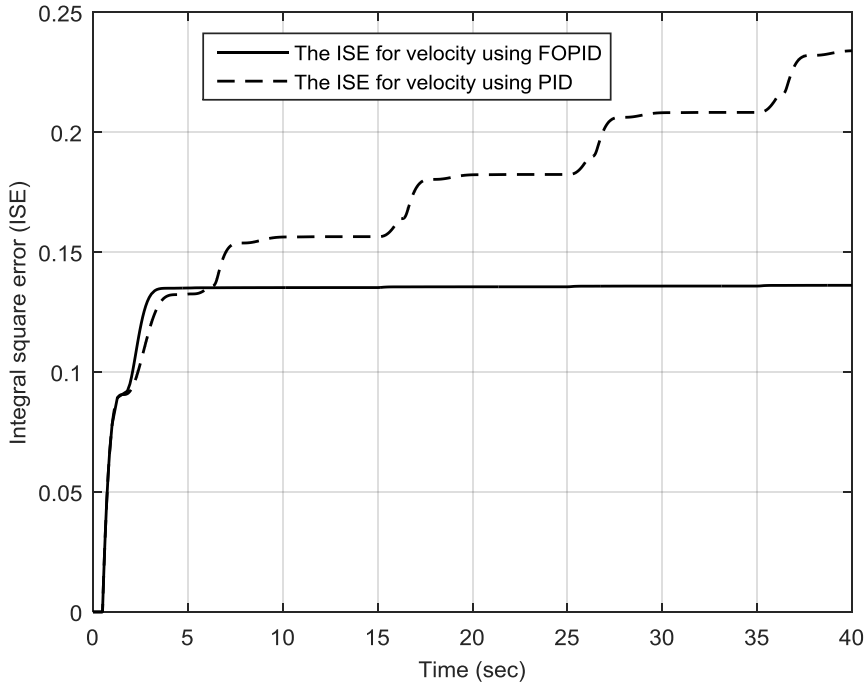



Fig. 4.61 ISE of velocity for square trajectory using PID and FOPID controllers.

## 4.8 Robustness Investigation

Designing an accurate and robust control system in the presence of disturbances is necessary for the effectiveness of control systems. Therefore, it is essential to establish robustness analysis of the fractional order to validate its performance in the presence of external disturbances. The source of such disturbances might be due to the frictions and roughness of

a ground surface. Hence, there is a necessity to verify that the design meets different operating conditions and should have a desirable robustness. The robustness of the fractional order PID controllers is examined by applying a disturbance due to an external friction that might be produced from the movement of the unmanned ground vehicle. To achieve this, two types of disturbances are carried out to verify functional operating conditions.

- Square pulses with different amplitudes,
- Sinusoidal signals with different amplitudes

The square pulses are applied as external disturbances that are associated with the input voltages of DC motors that are driving the left and right wheels. The amplitude of the square pulses varies from 0.1 volts to 1 volt; the frequency is fixed at 1 Hz. The duty cycle of the pulses is set to 10%. At each value, the cost function of the orientation can be calculated based on Eq. (4.10), given previously. Fig. 4.62 demonstrates the response of the cost function for the orientation against different amplitudes of square pulses for the linear, circular, lemniscate and square trajectories. In addition, Fig. 4.63 demonstrates the response of the cost function for the velocity under the same operating conditions; this is based on Eq. (4.11) given earlier. Moreover, the cost function of the four trajectories establishes that the circular trajectory has a minimum error rate. It is also observed that the square trajectory shows the maximum tracking error. This has occurred because of a non-continuous gradient that is inherent in the square trajectory.

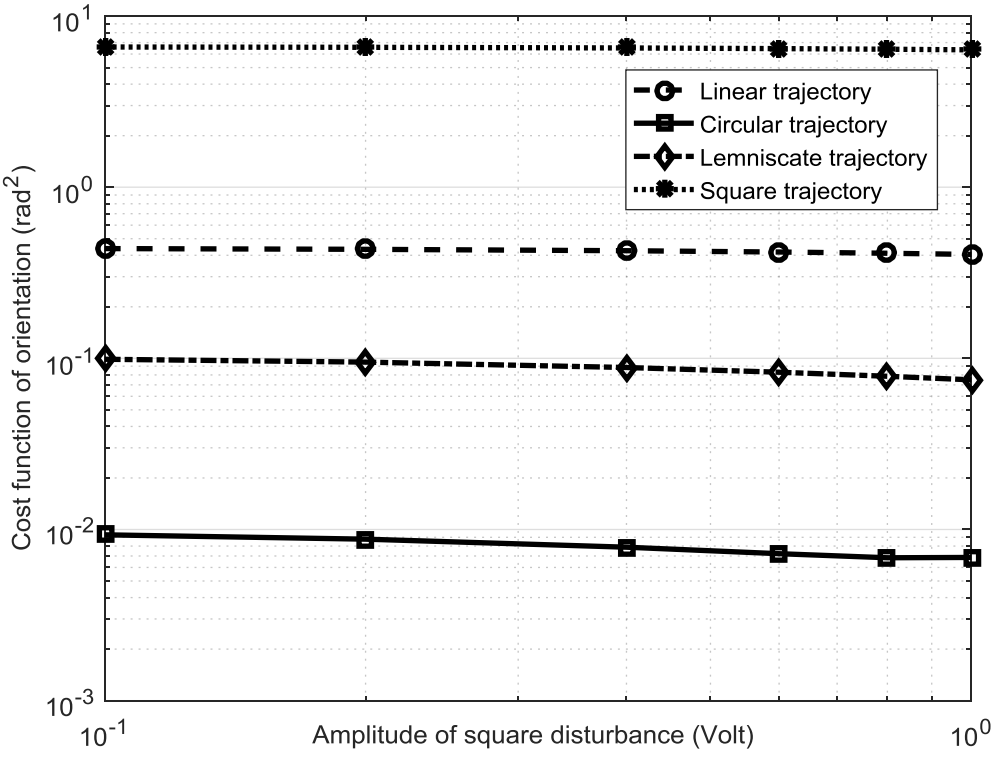



Fig. 4.62 Time response of cost function of orientation against square pulses.

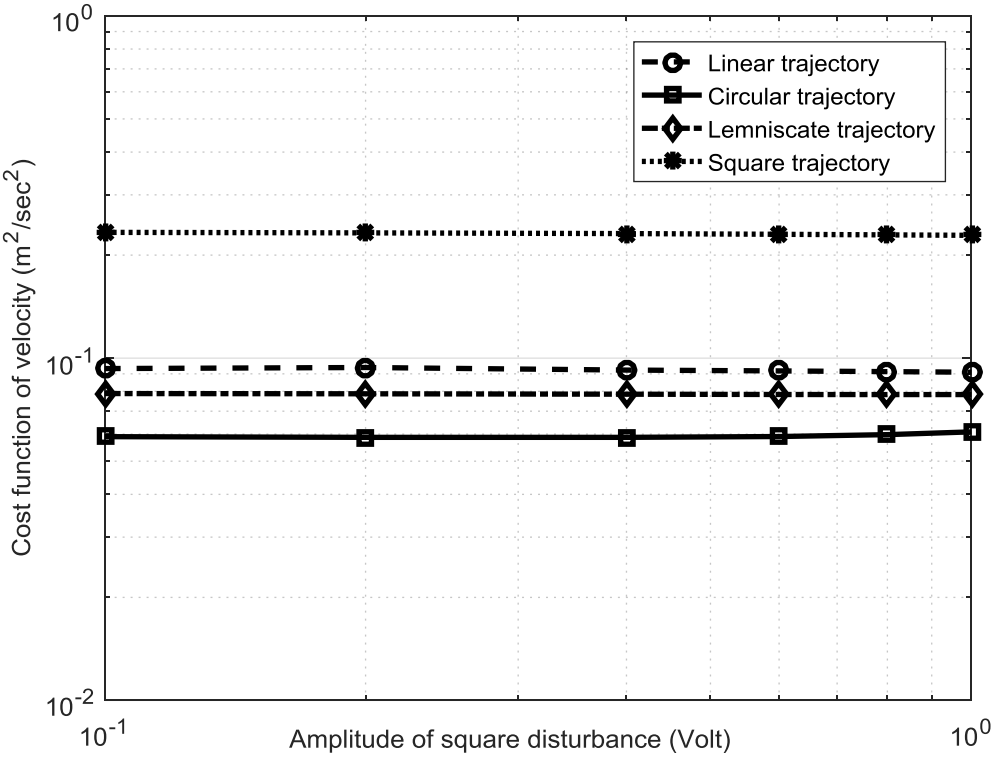



Fig. 4.63 Time response of cost function of velocity against square pulses.

Furthermore, a variety of sinusoidal signals based on different amplitudes have been applied and integrated with the actuators of the unmanned ground vehicle. The amplitude of the sinusoidal signals varies from 0.1volts to 1 volt. The angular frequency of all the sinusoidal signals is set at 1 rad/sec. The responses of the cost functions of the orientation and velocity against the changing of the amplitudes of the disturbances are shown in Fig. 4.64 and Fig. 4.65, respectively. It is noticeable that there is a slight changing in the response of the cost function when disturbances are applied. However, the influence of such a change is reasonable adequate and proves the robustness of the fractional order PID controller. It might be not appropriate when a considerable amount of disturbances is enforced. Hence, proper constraints have to be considered to avoid significant increases of disturbances. It is noticeable that the cost function for the square trajectory demonstrates the highest value, which in turn proves that the non-continuous gradient trajectory imposes challenge on the operation of the UGV. Therefore, a further improvement might be still needed to enhance the performance under such circumstances.

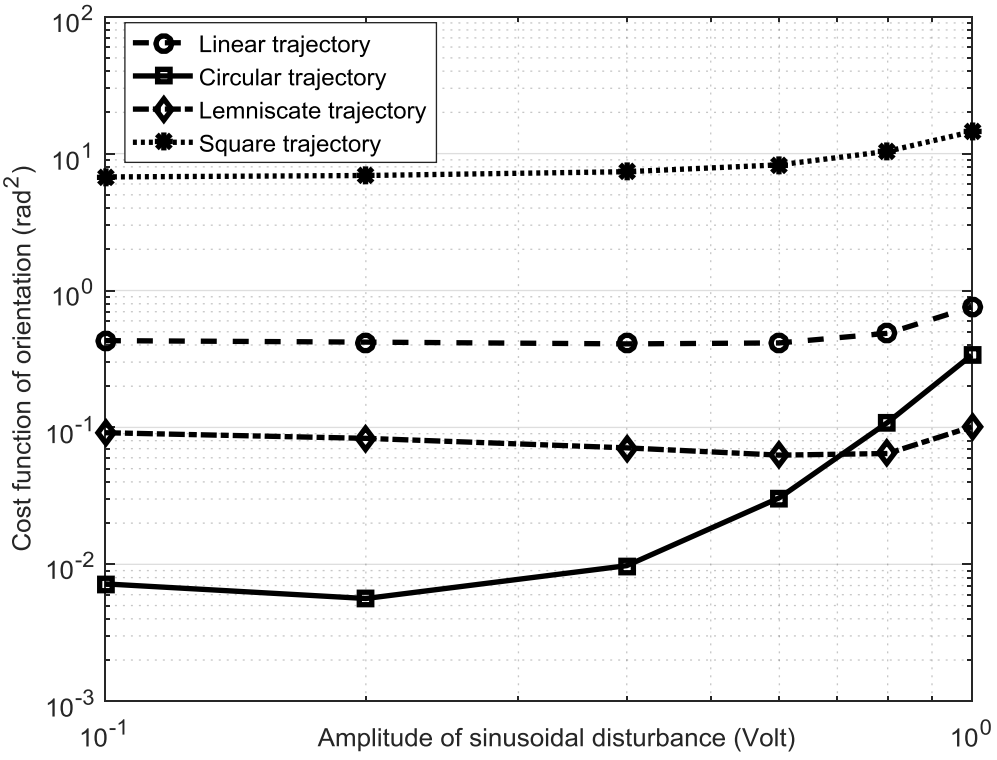



Fig. 4.64 Time response of cost function of orientation against sinusoidal signals.

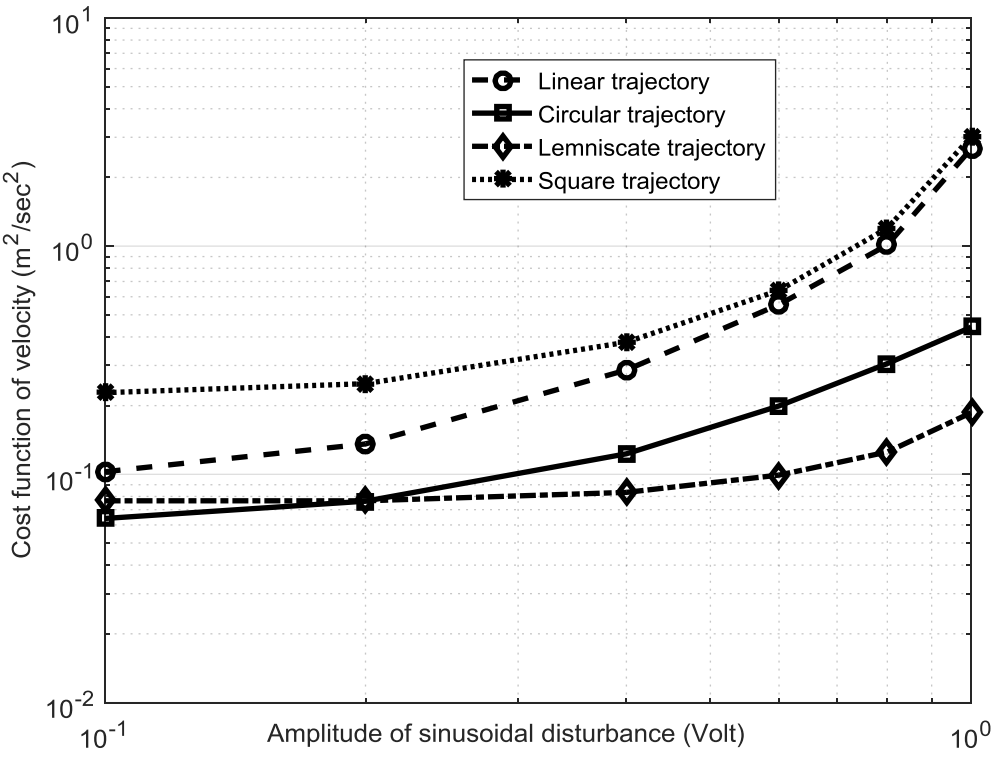



Fig. 4.65 Time response of cost function of velocity against sinusoidal signals.

## 4.9 Chapter Summary

The Fractional Order $PI^{\lambda}D^{\mu}$ controllers have been introduced to control the motion of the unmanned ground vehicle. The controller is optimized by minimizing the cost function based on a particle swarm optimisation algorithm. The PSO algorithm has been used to tune the parameters of the fractional order PID controller. The designed fractional order PID controllers have shown a significant ability to track different trajectories and the desired velocity. Four different trajectories are considered to validate the adaptation of the system. The simulation results have confirmed successfully the effectiveness and validation of the proposed fractional order PID controller in terms of minimising the trajectory tracking error and reducing control efforts. Additionally, the simulation results have proven the stability and robustness of the design.

The influence of external disturbances has been accounted to investigate the efficiency and robustness of the proposed fractional order PID controllers. The square pulses and sinusoidal signals are considered as two different sources of disturbances. These disturbances are integrated with the actuation of the UGV. It is observed that when disturbances are applied, a slight change in the cost function occurs. This change does not affect the vehicle operation even when it is applied persistently. The results have shown the effectiveness and robustness of the proposed fractional order PID controllers. In addition, the results have proven that the proposed methodology has reduced the trajectory tracking error significantly to a minimum and the system is globally asymptotically stable. Whereas, the trajectory tracking error of the square trajectory has not demonstrated a reasonable response and it is still noticeably high. Hence, a new control methodology is proposed in the next chapter by combining the fractional order PID controller and neural networks to minimise the trajectory-tracking error.

# Chapter 5

# Trajectory Tracking of UGV Based on NN-FOPID Controller

## 5.1 Introduction

This chapter contextualises the research by introducing background information on the application of neural networks (NN). They can be utilised to solve complex linear and nonlinear control problems. The NN has been widely used in a variety of dynamic systems. A multilayer neural network is capable of learning and identifying the characteristics of dynamic systems (Nguyen and Widrow, 1990). Early studies proposed NNs to tackle some basic applications. For example, Khalid and Omatu (1992) introduced a backpropagation neural network to learn the inverse dynamics model of a temperature control system. It had been proved that the ability of the neural network to learn the inverse kinematic of the process plant based on input vectors without a priori knowledge regarding the dynamics. The reported results were compared to the conventional proportional integral (PI) controller to demonstrate the advantages of NN.

The applications of neural networks to control systems have become increasingly broad. For instance, the application of neural networks was presented to control a double inverted pendulum (Arbo et al., 2014). Another application was proposed for neural networks of a thermal error compensation in computer numerical control machines (Nie, 2011). Recently, the neural networks have been applied extensively in a field of machine learning. In particular, many applications have been reported for a prediction and a deep learning based on neural networks. For instance, Rosli et al. (2016) presented different neural network architectures for developing a prediction model for gas metering systems. The deep learning approaches, which are implemented based on large neural networks, have become currently a major research area. One of the main applications of the deep learning is visual perception for the UGV.

## 5.2 Chapter Organisation

The chapter is organised as follows: The following section introduces a review of the architecture of neural networks. This also includes reviewing the comparisons among several optimisation algorithms that are utilised for tuning the parameters of neural networks. In

Section 5.4, the proposed design of the neural networks is presented. The conducted simulation results are described in Section 5.5 and the study is finally concluded in Section 5.6.

## 5.3 Neural Networks Architecture

The architecture of NN can be created using two or more combined neurons to form a multi-layer network (Fausett, 1993). Fig. 5.1 depicts a typical example of a multilayer architecture for a neural network. It is apparent that the architecture of an artificial neural network consists of three layers, i.e., input layer, hidden layer and output layer. First, the input layer receives variables related to a problem, which has a finite number of inputs and duplicate the value to their multiple outputs. The nodes of the input layer are passive. It means that they do not modify the data. The second layer for this example is the hidden layer that processes the information between the input and output layers of a network to develop a behavioural representation of the problem. Finally, the output layer provides the desired output of a trained system. There is a node given at the end of each layer. Each node represents a processing element that is active in comparison with the nodes in the input layer. The relationship between the nodes is manipulated with weights associated with nodes' outputs. This means that each node represents a summation value of all inputs that feed a particular node. Several transfer functions can be used to manipulate the relationship between the input and output of each node such as Sigmoid, Gaussian. In addition, there are biases associated with nodes to activate them.

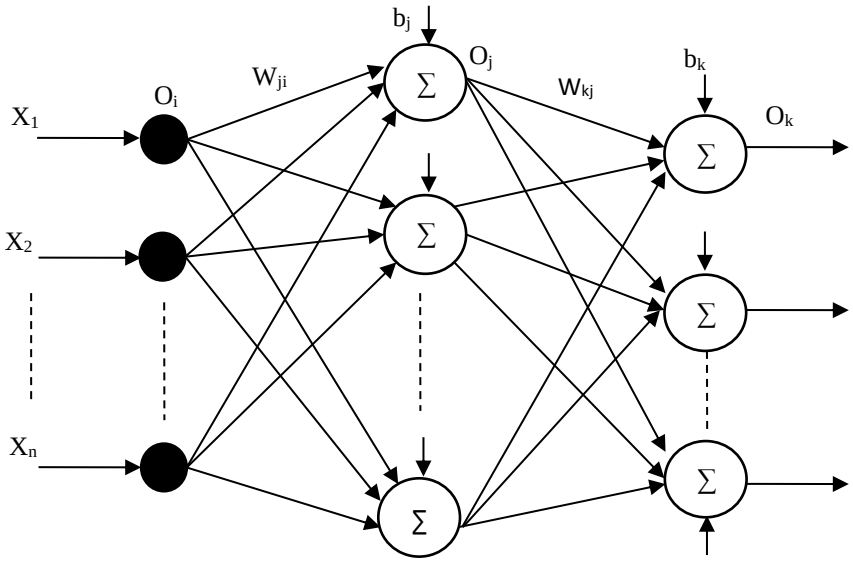



Fig. 5.1 Three-layer neural network.

The set of relationships for manipulating the interconnection between the layers at each stage is given below:

$$O_k = f(net_k) \tag{5.1}$$

$$net_k = \left( \sum_j W_{kj} O_j + b_k \right) \tag{5.2}$$

$$O_j = f(net_j) \tag{5.3}$$

$$net_j = \left( \sum_i W_{ji} O_i + b_j \right) \tag{5.4}$$

where

$W_{ji}$ - weights between the input layer and the hidden layer.

$W_{kj}$ - weights between the hidden layer and the output layer.

$b_j \; and \; b_k$ -biases of the hidden layer and the output layer, respectively.

In Equations (5.1) and (5.2), $f(net)$ is the transfer functions in both the hidden and output layer and can have different forms such as linear, sigmoid and hyperbolic tangent sigmoid transfer function. Transfer functions calculate a layer's output from its net input.

Different optimizations algorithms have already been developed for neural-networks training. For instance, the back-propagation (BP) algorithm that could be considered one of the most applied algorithms for training of ANN (Wilamowski, 2009). Currently, the BP algorithm is still widely for optimisation purposes. However, the slow convergence makes this algorithm to be considered an inefficient algorithm. The two main causes of the slow convergence in BP algorithm are; its step sizes and the curvature of the error surface may not be the same in all directions. Gauss-Newton algorithm is introduced as new algorithm to greatly improve the slow convergence. This algorithm is based on second-order derivatives of an error function to assess the error in the curvature surface in contrast with BP algorithm, which is based on first order derivative.

The step size in the Gauss–Newton algorithm can be found for each direction that will converge speedily. In particular, if the error function has a quadratic surface. However, there is still a problem occurs if the quadratic approximation of error function is not reasonable. This in turn will lead the Gauss–Newton algorithm to be mostly divergent (Wilamowski and Irwin, 2011). Therefore, the neural is introduced due to its benefits over the BP and Gauss–Newton algorithms. Levenberg-Marquardt (LM) algorithm is a combination of BP algorithm and Gauss–Newton algorithm. In LM algorithm, a numerical solution is provided to a problem for minimizing a nonlinear function. Moreover, it is fast and has stable convergence, suitable for training small and medium sized problems. It inherits the stability of the BP algorithm and the

speed advantage of the Gauss–Newton algorithm. To fully understand the derivation of the LM algorithm, the following four training algorithms will be presented; beginning with (1) Back-propagation algorithm, (2) Newton's method, (3) Gauss–Newton's algorithm, and ending with (4) Levenberg–Marquardt algorithm.

The mean square error (MSE) is defined in the equation below to evaluate the error value in training process. It is calculated for all training process and network outputs as follows:

$$E(x,w) = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M} e_{p,m}^2 \tag{5.5}$$

$$\mathrm{e_{p,m}} = \mathrm{d_{p,m}} - \mathrm{o_{p,m}} \tag{5.6}$$

where

x - Network input,

w - Weight of network,

$p$ - Number of patterns,

$m$ - Number of outputs,

$\mathrm{e_{p,m}}$ - Training error,

d - Required output, and

o - Actual output.

### *5.3.1 Back-Propagation Algorithm*

The BP algorithm is used for finding the minimum of the error function. It utilises a gradient descent method to calculate the error in weight space to be a solution of the learning problem. Therefore, the error function can be minimized by using an iterative process of the gradient descent as shown in equation below. The index 'g' is defined as the first-order derivative of the total error function:

$$g = \frac{\partial E(x,w)}{\partial w} = \left[\frac{\partial E}{\partial w_1} \quad \frac{\partial E}{\partial w_2} \quad \dots \quad \frac{\partial E}{\partial w_N}\right]^T \tag{5.7}$$

The update rule for each weight of the BP algorithm could be written as follows:

$$w_{k+1} = w_k - \alpha g_k \tag{5.8}$$

where

$\alpha$ - Learning rate or it is called the step size,

$N$ - Number of weights,

$i$ and $j$ - Indices of weights, from 1 to $N$, and

$k$ - Iteration number.

### *5.3.1 Newton's Algorithm*

In Newton's method, it is assumed that all the gradient components, i.e., $g_1, g_2, \dots, g_N$ are the functions of weights where all weights are linearly independent:

$$\begin{cases} g_1 = F_1(w_1, w_2, \dots w_N) \\ g_2 = F_2(w_1, w_2, \dots w_N) \\ \dots \\ g_N = F_N(w_1, w_2, \dots w_N) \end{cases} \tag{5.9}$$

where $F_1, F_2, \dots$ and $F_N$ represent the nonlinear relationships between gradient components and weights. Thus, to unfold each $g_i$ $(i = 1, 2, \dots, N)$ in Eq. (5.9) by Taylor series and take the first-order approximation, it can be obtained:

$$\begin{cases} g_1 \approx g_{1,0} + \dfrac{\partial g_1}{\partial w_1}\Delta w_1 + \dfrac{\partial g_1}{\partial w_2}\Delta w_2 + \cdots + \dfrac{\partial g_1}{\partial w_N}\Delta w_N \\ g_2 \approx g_{2,0} + \dfrac{\partial g_2}{\partial w_1}\Delta w_1 + \dfrac{\partial g_2}{\partial w_2}\Delta w_2 + \cdots + \dfrac{\partial g_2}{\partial w_N}\Delta w_N \\ \dots \\ g_N \approx g_{N,0} + \dfrac{\partial g_N}{\partial w_1}\Delta w_1 + \dfrac{\partial g_N}{\partial w_2}\Delta w_2 + \cdots + \dfrac{\partial g_N}{\partial w_N}\Delta w_N \end{cases} \tag{5.10}$$

From the definition of the gradient descent $g$ in Eq. (5.7), it could be determined that

$$\frac{\partial g_i}{\partial w_j} = \frac{\partial(\frac{\partial E}{\partial w_i})}{\partial w_j} = \frac{\partial^2 E}{\partial w_i \partial w_j} \tag{5.11}$$

By substituting Eq. (5.7) into Eq. (5.10), we obtain:

$$\begin{cases} g_1 \approx g_{1,0} + \dfrac{\partial^2 E}{\partial w_1^2}\Delta w_1 + \dfrac{\partial E}{\partial w_1 \partial w_2}\Delta w_2 + \cdots + \dfrac{\partial E}{\partial w_1 \partial w_N}\Delta w_N \\ g_2 \approx g_{2,0} + \dfrac{\partial^2 E}{\partial w_2 \partial w_1}\Delta w_1 + \dfrac{\partial^2 E}{\partial w_2^2}\Delta w_2 + \cdots + \dfrac{\partial^2 E}{\partial w_2 \partial w_N}\Delta w_N \\ \dots \\ g_N \approx g_{N,0} + \dfrac{\partial^2 E}{\partial w_N \partial w_1}\Delta w_1 + \dfrac{\partial^2 E}{\partial w_N \partial w_2}\Delta w_2 + \cdots + \dfrac{\partial^2 E}{\partial w_N^2}\Delta w_{2N} \end{cases} \tag{5.12}$$

In order to obtain the minima of error function, the gradient descent should be zero of each component. Therefore, left side of the Eq. (5.12) are all set to zero, hence

$$\begin{cases} 0 \approx g_{1,0} + \dfrac{\partial^2 E}{\partial w_1^2}\Delta w_1 + \dfrac{\partial^2 E}{\partial w_1 \partial w_2}\Delta w_1 + \cdots + \dfrac{\partial^2 E}{\partial w_1 \partial w_N}\Delta w_N \\ 0 \approx g_{2,0} + \dfrac{\partial^2 E}{\partial w_2 \partial w_1}\Delta w_1 + \dfrac{\partial^2 E}{\partial w_2^2}\Delta w_2 + \cdots + \dfrac{\partial^2 E}{\partial w_2 \partial w_N}\Delta w_N \\ \dots \\ 0 \approx g_{N,0} + \dfrac{\partial^2 E}{\partial w_N \partial w_1}\Delta w_1 + \dfrac{\partial^2 E}{\partial w_N \partial w_2}\Delta w_2 + \cdots + \dfrac{\partial^2 E}{\partial w_N^2}\Delta w_{2N} \end{cases} \tag{5.13}$$

By combining Eq. (5.7) with (5.13), it yields

$$\begin{cases} -\dfrac{\partial E}{\partial w_1} = -g_{1,0} \approx +\dfrac{\partial^2 E}{\partial w_1^2}\Delta w_1 + \dfrac{\partial^2 E}{\partial w_1 \partial w_2}\Delta w_1 + \cdots + \dfrac{\partial^2 E}{\partial w_1 \partial w_N}\Delta w_N \\ -\dfrac{\partial E}{\partial w_2} = -g_{2,0} \approx \dfrac{\partial^2 E}{\partial w_2 \partial w_1}\Delta w_1 + \dfrac{\partial^2 E}{\partial w_2^2}\Delta w_2 + \cdots + \dfrac{\partial^2 E}{\partial w_2 \partial w_N}\Delta w_N \\ \cdots \\ -\dfrac{\partial E}{\partial w_N} = -g_{N,0} \approx \dfrac{\partial^2 E}{\partial w_N \partial w_1}\Delta w_1 + \dfrac{\partial^2 E}{\partial w_N \partial w_2}\Delta w_2 + \cdots + \dfrac{\partial^2 E}{\partial w_N^2}\Delta w_{2N} \end{cases} \tag{5.14}$$

From the equation above, it is obvious that there are N parameters for N equations. This means all $\Delta w_i$ can be calculated. During the learning process, the weights will be updated iteratively. Eq. (5.14) can be also written in a matrix form as follows:

$$\begin{bmatrix} -g_1 \\ -g_2 \\ \cdots \\ -g_N \end{bmatrix} = \begin{bmatrix} -\dfrac{\partial E}{\partial w_2} \\ -\dfrac{\partial E}{\partial w_1} \\ \cdots \\ -\dfrac{\partial E}{\partial w_N} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial^2 E}{\partial w_1^2} & \dfrac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \dfrac{\partial^2 E}{\partial w_1 \partial w_N} \\ \dfrac{\partial^2 E}{\partial w_2 \partial w_1} & \dfrac{\partial^2 E}{\partial w_2^2} & \cdots & \dfrac{\partial^2 E}{\partial w_2 \partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\partial^2 E}{\partial w_N \partial w_1} & \dfrac{\partial^2 E}{\partial w_N \partial w_2}\Delta w_2 & \cdots & \dfrac{\partial^2 E}{\partial w_N^2} \end{bmatrix} \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \cdots \\ \Delta w_N \end{bmatrix} \tag{5.15}$$

where, the square matrix is Hessian matrix:

$$H = \begin{bmatrix} \dfrac{\partial^2 E}{\partial w_1^2} & \dfrac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \dfrac{\partial^2 E}{\partial w_1 \partial w_N} \\ \dfrac{\partial^2 E}{\partial w_2 \partial w_1} & \dfrac{\partial^2 E}{\partial w_2^2} & \cdots & \dfrac{\partial^2 E}{\partial w_2 \partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\partial^2 E}{\partial w_N \partial w_1} & \dfrac{\partial^2 E}{\partial w_N \partial w_2}\Delta w_2 & \cdots & \dfrac{\partial^2 E}{\partial w_N^2} \end{bmatrix} \tag{5.16}$$

By combining Equations (5.7) and (5.16) with Eq. (5.15)

$$g = -H\Delta w \tag{5.17}$$

Then $$\Delta w = -H^{-1} g \tag{5.18}$$

In addition, in Newton's method, the incremental updating rule for weights can be given below:

$$w_{k+1} = w_k - H_k^{-1} g_k \tag{5.19}$$

where H is defined as a Hessian matrix, which provides the second-order derivatives of total error function and gives a proper evaluation of the change of the gradient descent. By comparing Equations (5.18) and (5.19), it may be noticed that well matched step sizes are given by the inverted Hessian matrix.

### *5.3.3 Gauss-Newton Algorithm*

In Gauss Newton algorithm, Jacobian matrix J is introduced to simplify the calculation process due to the complexity inherited in the second-order derivatives of total error function with Newton's method.

$$J = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \dots & \frac{\partial e_{1,1}}{\partial w_N} \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & \dots & \frac{\partial e_{1,2}}{\partial w_N} \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_{p,1}}{\partial w_1} & \frac{\partial e_{p,1}}{\partial w_2} & \dots & \frac{\partial e_{p,1}}{\partial w_N} \\ \frac{\partial e_{p,2}}{\partial w_1} & \frac{\partial e_{p,2}}{\partial w_2} & \dots & \frac{\partial e_{p,2}}{\partial w_N} \\ \dots & \dots & & \dots \\ \frac{\partial e_{p,m}}{\partial w_1} & \frac{\partial e_{p,m}}{\partial w_2} & & \frac{\partial e_{p,m}}{\partial w_N} \end{bmatrix} \tag{5.20}$$

By integrating Equations (5.5) and (5.7), gradient descent's elements can be calculated as follows:

$$g_i = \frac{\partial E}{\partial w_i} = \frac{\partial (\frac{1}{2}\sum_{p=1}^{p}\sum_{m=1}^{m} e_{p,m}^2)}{\partial w_i} = \sum_{p=1}^{p}\sum_{m=1}^{m}\left(\frac{\partial e_{p,m}}{\partial w_i}\, e_{p,m}\right) \tag{5.21}$$

Combining Equations (5.20) and (5.21), the relationship between gradient descent (g) and Jacobian matrix (J) would be:

$$g = J.e \tag{5.22}$$

where the error (e) has the following form:

$$e = \begin{bmatrix} e_{1,1} \\ e_{1,2} \\ \dots \\ e_{1,m} \\ \dots \\ e_{p,1} \\ e_{p,1} \\ \dots \\ e_{p,m} \end{bmatrix} \tag{5.23}$$

By inserting Eq. (5.5) into Eq. (5.16), the elements of the Hessian matrix, i.e., i$^{th}$ row and j$^{th}$ column can be calculated as:

$$h_{i,j} = \frac{\partial^2 E}{\partial w_i \partial w_j} = \frac{\partial^2 (\frac{1}{2}\sum_{p=1}^{p}\sum_{m=1}^{m} e_{p,m}^2)}{\partial w_i \partial w_j} = \sum_{p=1}^{p}\sum_{m=1}^{m}\frac{\partial e_{p,m}}{\partial w_i}\frac{\partial e_{p,m}}{\partial w_j} + S_{i,j} \tag{5.24}$$

where $S_{i,j} = \sum_{p=1}^{p} \sum_{m=1}^{m} \frac{\partial^2 e_{p,m}}{\partial w_i \partial w_j}\ e_{p,m}$ (5.25)

From Newton's method, it is assumed that the $S_{i,j}$ approaches to zero. Therefore, the relationship between Jacobian matrix (J) and Hessian matrix (H) can be rewritten as follows:

$$H \approx J^T J \tag{5.26}$$

By combining Equations (5.19), (5.22), and (5.26), the weights that update rule of the Gauss–Newton algorithm can be given as in below:

$$w_{k+1} = w_k - (J_k^T J_k)^{-1}\ J_k e_k \tag{5.27}$$

### *5.3.4 Levenberg-Marquardt Algorithm*

This algorithm is an approximation to Newton's method (Hagan and Menhaj, 1994). In order to make sure that the approximated Hessian matrix is invertible, LM algorithm introduces another approximation to Hessian matrix as follows:

$$H \approx J^T J + \mu I \tag{5.28}$$

where

μ = combination coefficient and it is always positive,

I = the identity matrix.

By combining Equations (5.27) and (5.28), the update rule for weights of LM algorithm can be presented as follows:

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1}\ J_k e_k \tag{5.29}$$

The LM algorithm switches between the backpropagation algorithms and the Gauss–Newton algorithm during the training process. Two situations will be considered in LM algorithm. Firstly, if the combination coefficient μ is quite small, hence, Eq. (5.29) is approaching to Eq. (5.27) and Gauss–Newton algorithm is used. However, if the combination coefficient μ is quite large, Eq. (5.26) approximates to Eq. (5.8) and the BP algorithm is used.

The following steps are described the training process of LM algorithms:

**Step 1:** Generate the initial weights,

**Step 2:** update weights using Eq. (5.29),

**Step 3:** Evaluate the error at each updated weights,

**Step 4:** If the new error is increased after updating, then go to **step 2** and try an update again after increasing combination coefficient $\mu$ by a suitable factor. Otherwise, go to **step 5**,

**Step 5:** If the new error is decreased. Compare the new error with the required value. If the new error is smaller than the required value, then stop learning. Otherwise, go to **step 2**.

The differences among the aforementioned algorithms are highlighted Table 5.1. It summarizes the update rules, convergence and computation complexity of each optimization algorithm.

Table 5.1 Specifications of the various algorithms.

| **Algorithms** | **Update Rules** | **Convergence** | **Computation Complexity** |
|---|---|---|---|
| BP algorithm | $w_{k+1} = w_k - \alpha g_k$ | Stable, slow | Gradient |
| Newton algorithm | $w_{k+1} = w_k - H_k^{-1} g_k$ | Unstable, fast | Gradient and Hessian |
| Gauss–Newton algorithm | $w_{k+1} = w_k - (J_k^T J_k)^{-1}\ J_k e_k$ | Unstable, fast | Jacobian |
| Levenberg-Marquardt Algorithm | $w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1}\ J_k e_k$ | stable, fast | Jacobian |

## 5.4 Design of Neural Networks

The architecture of NN is reviewed previously in Section 5.3. In reminder of this chapter, the aim is to describe the proposed design of NN based on the fractional order PID controller. An intelligent control system can be implemented based on artificial neural networks to control the trajectory tracking of UGV. The implementation process of the control scheme for the neural networks can be achieved by building, training and validating the NN. It is needed to define and initiate the main parameters of NN. Multi-layers are embedded with each other via synapse matrices. Each synapse has a random weight assigned to it. In each layer of ANNs, there is a transfer function i.e. Sigmoid function. Such a function runs in every neuron of a network when data is supplied.

Sigmoid function, $A(x) = \frac{1}{1+e^{-x}}$ (5.30)

The architecture of the neural networks is presented based on that fractional order PID controller as depicted in Fig. 5.2. It represents a combination of the neural network and the fractional order PID controller, abbreviated as NN-FOPID. The input of this combination is the error signal between the actual output and the desired input. These error signals are governed by the proposed NN-FOPID controller to produce suitable output signals, which they drive actuators of a UGV to follow pre-defined trajectories.

The main functions of the neural networks in this structure are to build a system, which is capable of tracking the movement of a UGV based on measuring the errors between the actual and desired movements. The neural networks utilise their adaptive and learning capabilities to learn and predict the best needed control actions based on the available data. The datasets of the errors measurements are taken from the previously designed FOPID controllers. The measured errors are processed based on the three stages i.e. input, hidden and output layers. For instance, in the input layer, the measured errors become the input of the neural network that are further transformed using sigmoid transfer function given in Equation (5.30). This function transfers non-linear errors to linear error and it will limit the range of measured error between '0' and '1'. Then, the samples can be fed into the hidden layer. Similarly, the outputs of the hidden layer are weighted using the Equation (5.4) and it is again shown in the neural networks block below. The latter equation is existed between the input and hidden layers and between the hidden and output layers. It calculates a weighted sum of its input, adds a bias in order to process into the next stage.

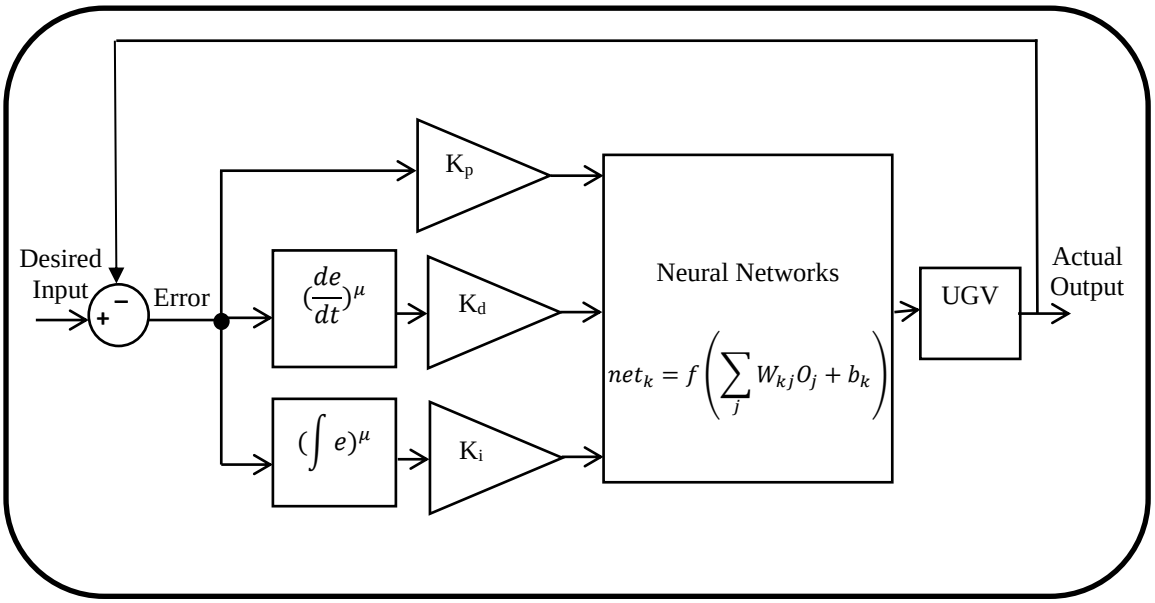



Fig. 5.2 Neural network of one part of UGV model.

Fig. 5.3 demonstrates the training phase for which NN-FOPID controller using the LM training algorithm. The network training function given in Equation (5.29) earlier updates weight and bias values according to Levenberg-Marquardt optimization until obtaining the overall performance is optimised and the error rate is minimised. The final layout of the control architecture and UGV is depicted in Fig. 5.4. Two trained neural network controllers are used

for controlling the voltages of the right and left actuators. This enables the vehicle to adjust its heading and consequently track a predefined trajectory based on the guidance of its wheels. The first controller receives the error between the desired generated trajectory and actual trajectory in order to control the orientation angle of the UGV. Therefore, the vehicle must change its orientation as needed to track the desired trajectory. The output of this controller is directly connected to the right motor.

The second controller utilises the error signal between the desired and actual velocities. The desired velocity is assumed to a constant value during the tracking process. The output of this controller is fed to the left motor voltage of the UGV. The main purpose of the second controller is to maintain a constant velocity for controlling the motion. The input and output data of NN are obtained from the FOPID controller that implemented in Chapter 4. They are used to train the parameters of the neural controller using the LM training algorithm. The optimal values of the trainable parameters of the neural controller are obtained using a cost function based on a mean square error.

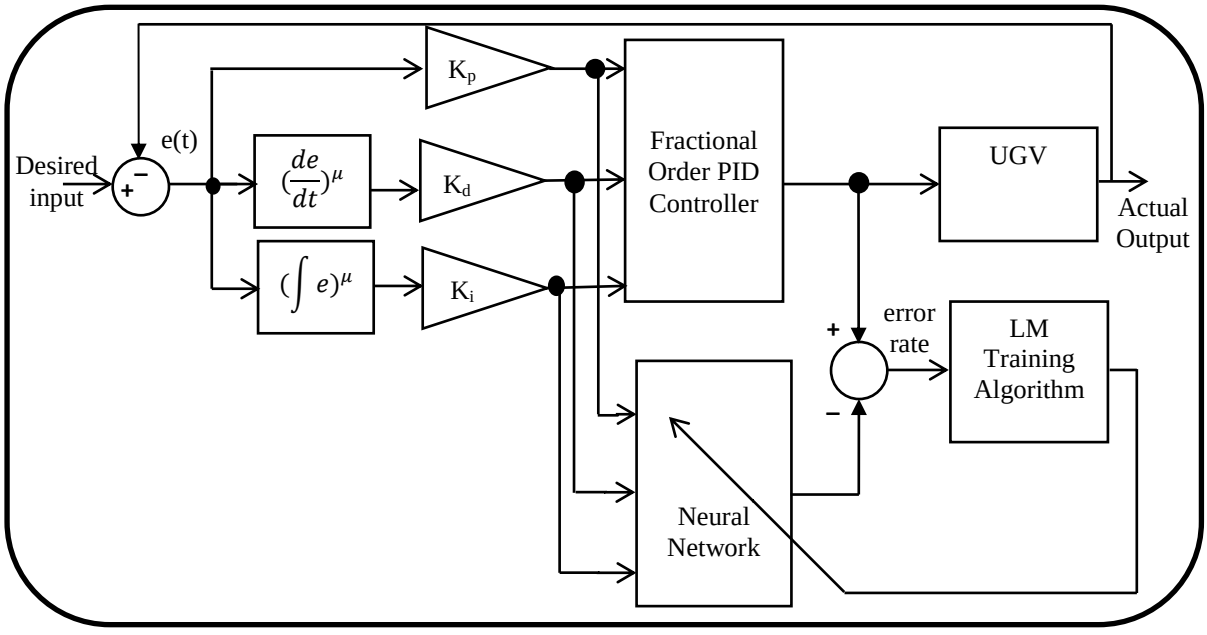



Fig. 5.3 Training phase of neural networks using LM training algorithm.

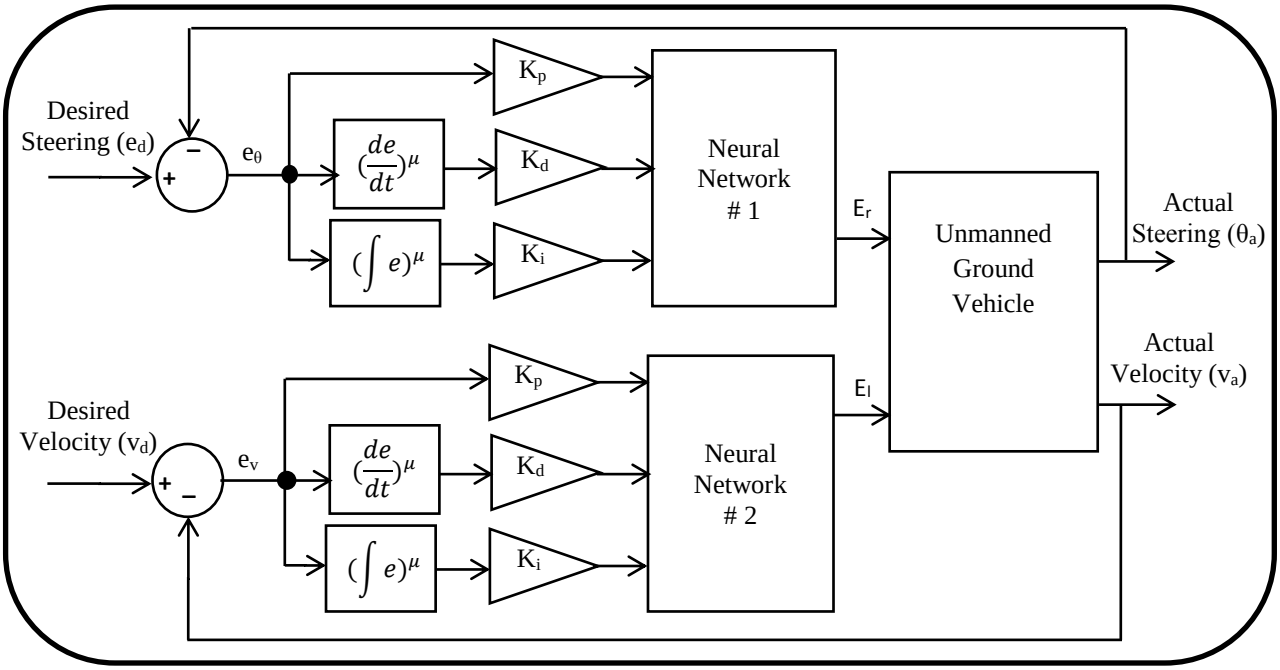



Fig. 5.4 Block diagram of the neural networks and UGV.

The tracking orientation error and tracking velocity error are measured by the equations below, respectively:

$$e_\theta(t) = \theta_d(t) - \theta_a(t) \tag{5.31}$$

$$e_v(t) = v_d(t) - v_a(t) \tag{5.32}$$

where

$\theta_d(t)$ - Desired orientation angle,

$\theta_a(t)$ - Actual orientation angle,

$e_\theta(t)$ - Tracking orientation error,

$v_r(t)$ - Desired velocity,

$v_a(t)$ - Actual velocity, and

$e_v(t)$ - Tracking velocity error.

The parameters of neural network # 1 are discussed as follows: The number of neurons in the hidden layer is seven. The type of the transfer function used in the hidden layer is hyperbolic tangent sigmoid transfer function. It means that the number of biases in the hidden layer is seven, and the number of weights between the input layer and the hidden layer equals twenty-one. Because of there is only one output for each NN controller, it means the number of weights between the hidden layer and the output layer is seven and there is only one set of biases in the

output layer. A linear transfer function is used in the output layer. The weights and biases of this network are given below.

The performance progress of training against epoch numbers is depicted in Fig. 5.5. It is observable that the mean squared error equals $1.3628 \times 10^{-05}$, which is the minimum average squared error and the best training performance obtained between outputs and targets. The dashed line is the best goal that equals $10^{-05}$ as set in the MATLAB code. In Fig. 5.6, the dashed line represents the perfect result (output=target). The solid line represents the best-fit linear regression line between outputs and targets. The value of the parameter R is an indication of the relationship between the outputs and targets. If $R$ = 1, this indicates that there is an exact linear relationship between outputs and targets. If $R$ is close to zero, then there is no linear relationship between outputs and targets. It is observable that $R$=1 which it indicates the exact linear relationship as desired.

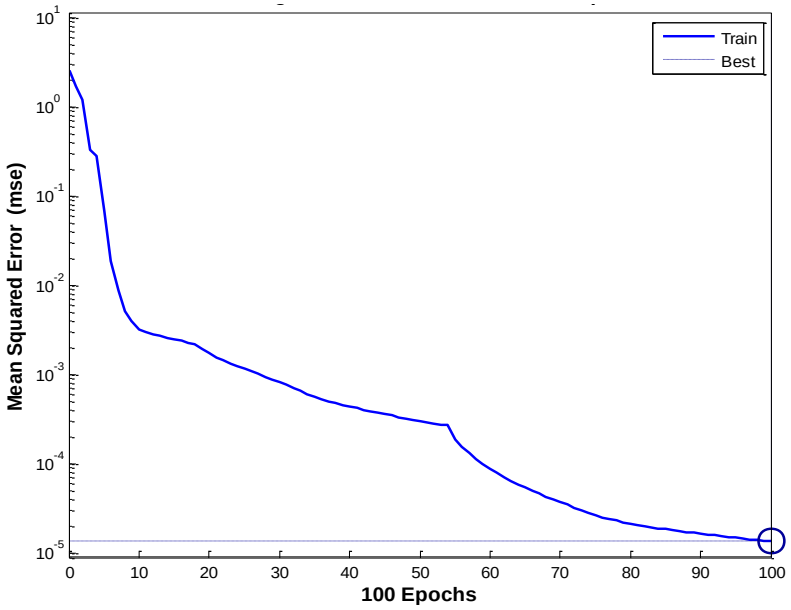



Fig. 5.5 Training performance for NN of orientation tracking control.

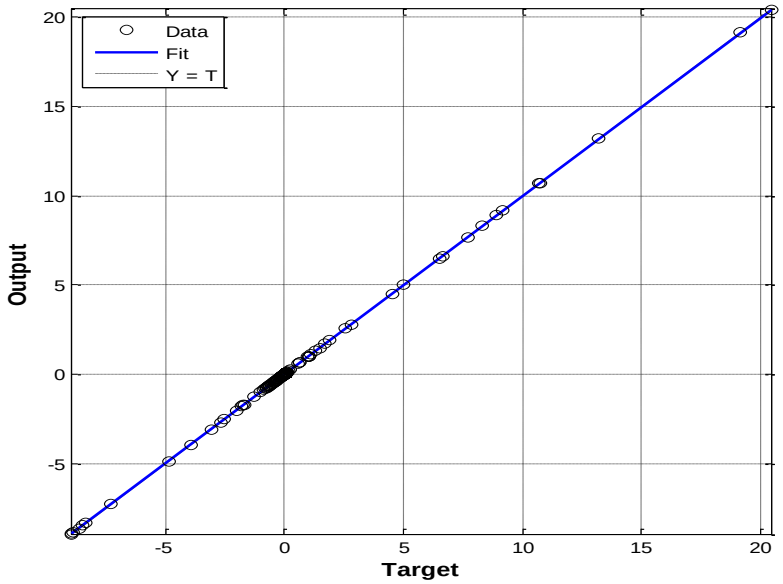



Fig. 5.6 Regression plot for NN of orientation tracking control.

Similarly, the parameters of neural network # 2 stated that the best training performance based on mean squared error equals $2.3545 \times 10^{-05}$ as depicted in Fig. 5.7. In this network, the number of neurons in the hidden layer is ten. Linear and hyperbolic tangent sigmoid transfer function are used in the hidden and output layers respectively. In Fig. 5.8, it is noticeable that $R$=1, which indicates the exact linear relationship as targeted.

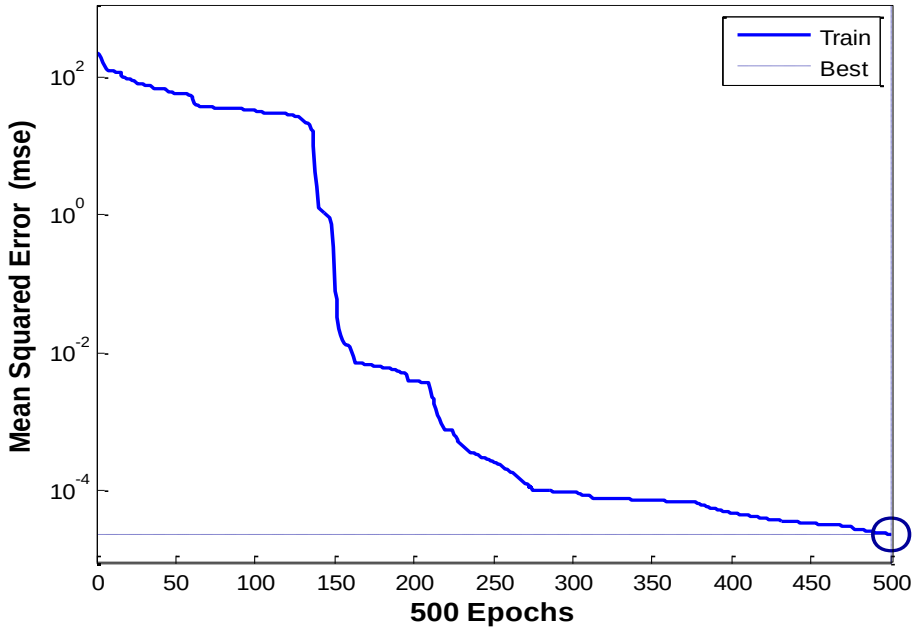



Fig. 5.7 Training performance for NN of velocity tracking control.

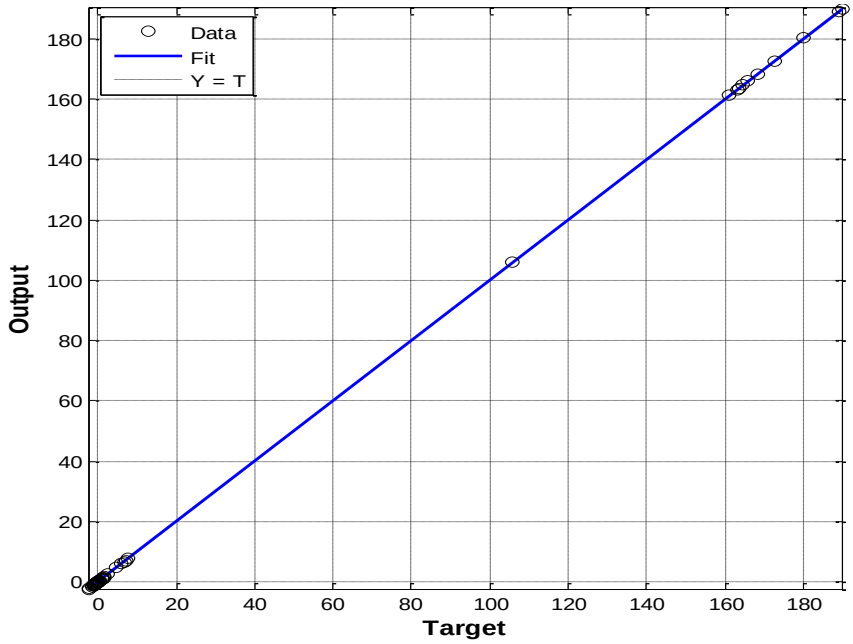



Fig. 5.8 Regression plot for NN of velocity tracking control.

## 5.5 Simulation results

In this section, the simulation results of the applying the artificial neural networks based on the fractional order PID controller are presented for solving trajectory tracking problem in the UGV. The NN controllers are optimised by Levenberg-Marquardt algorithm. The introduced technique shows a remarkable improvement in terms of minimizing trajectory-tracking error in comparison with the other controllers. The architecture of NN consists of two neural controllers. The first one deals with steering control to enforce the UGV tracking of the give trajectory. Whereas, the second NN deals with tracking a reference velocity to maintain a constant velocity during the movement. The parameters of these two NN controllers are obtained using Levenberg-Marquardt algorithm, i.e., weights and biases.

The outcome results are compared with FOPID controller introduced earlier in Chapter 4. Despite the progress in the findings made previously, it is demonstrated an effective way of minimising the trajectory tracking error for the continuous gradient trajectories. However, the simulation results presented an insufficient tracking error for the non-continuous gradient trajectory i.e. square trajectory. Therefore, this case reveals the need for further investigation into the non-continuous gradient trajectory in particular to minimise the trajectory tracking error, hence, to improve and enhance the performance of trajectory tracking. Other significant aspect is to minimise the orientation tracking error and eradicate from the delay in the time

response. A comparison of the results for three control methodologies reveals that the trajectory tracking error has been improved significantly as illustrated in Fig. 5.9.

It is noticeable that the UGV has tracked the desired square trajectory in a quite efficient manner using neural networks that have been developed based on the fractional order PID controller. In addition, the differences in the orientations of the UGV based on the three control methodologies are introduced in Fig. 5.10. The simulation results obtained from the NN controller show that the orientation angle converges faster to the corresponding orientation angle by comparison with the traditional and fractional order PID controllers. These findings have significant implications for minimising the error of the orientation that is shown in Fig. 5.11. Moreover, these findings enhance the understanding of the predication capability and substantial adaptation of the neural networks when they have been associated with the fractional order PID controller. Furthermore, the neural networks based on fractional order PID controller make a significant contribution to the non-continuous gradient square trajectory and have approved the generalisability for the both of continuous gradient and non-continuous gradient trajectories.

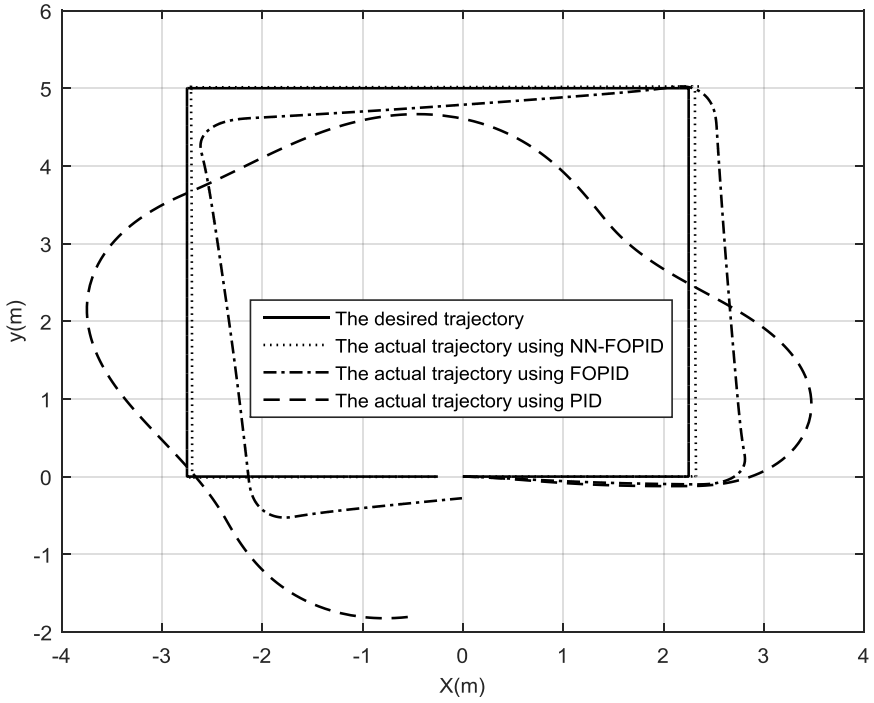



Fig. 5.9 Square trajectories using the PID, FOPID and NN-FOPID controllers.

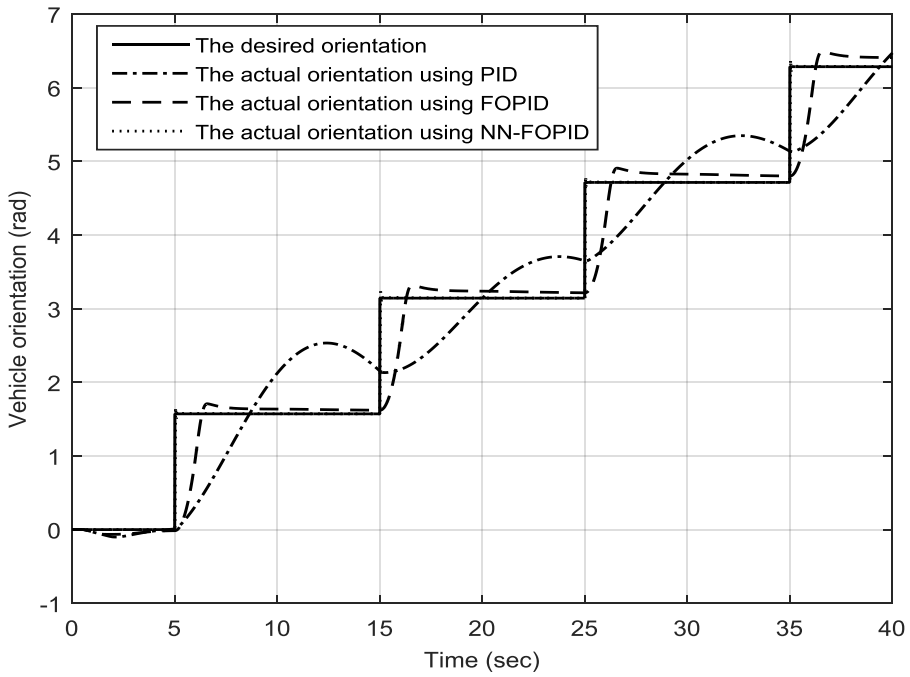



Fig. 5.10 Orientations for square trajectory using PID, FOPID and NN-FOPID controllers.

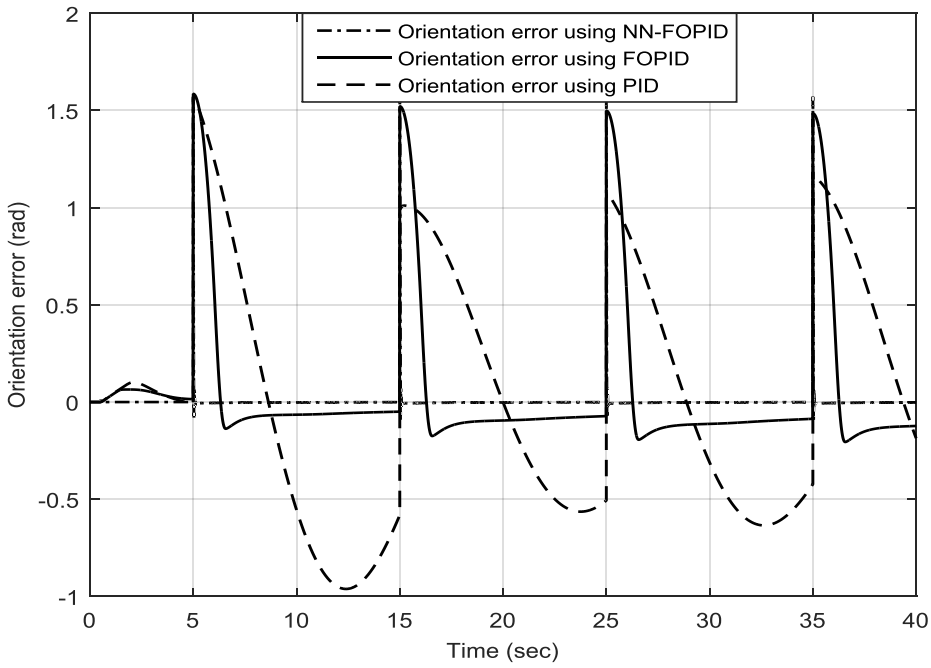



Fig. 5.11 Error in orientation for square trajectory using PID, FOPID and NN-FOPID controllers.

To investigate and demonstrate the effectiveness of the NN-FOPID controllers, Figs. 4.12 and 5.13 are introduced to compare the control efforts for the left wheel and right wheel, respectively. There are no significant differences between the control efforts obtained by

FOPID and NN-FOPID controllers. Nonetheless, a slight improvement is observed at the corners of changing the gradient of the non-continuous square trajectory.

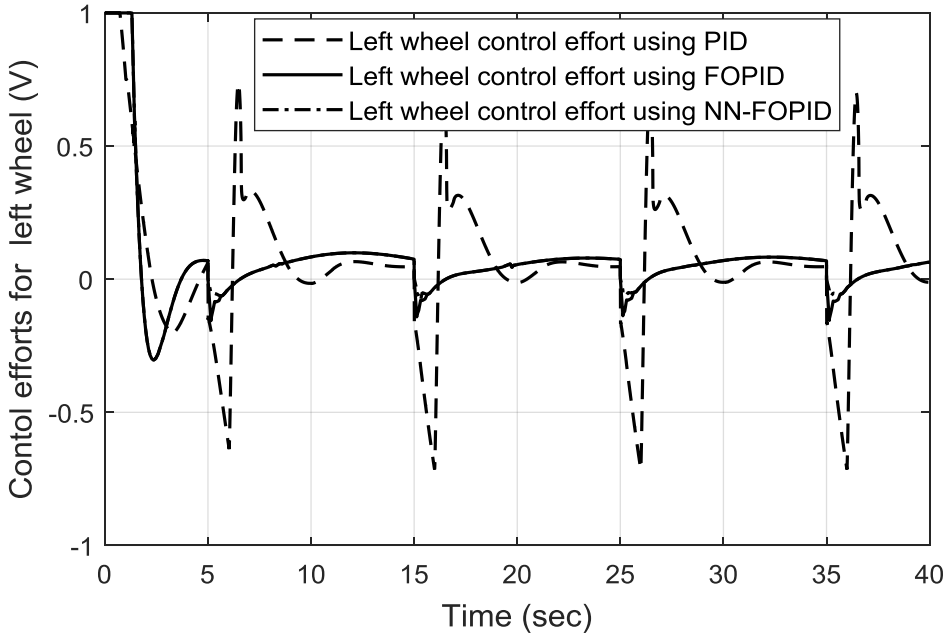



Fig. 5.12 Control efforts for left wheel of square trajectory using PID, FOPID and NN-FOPID controllers.

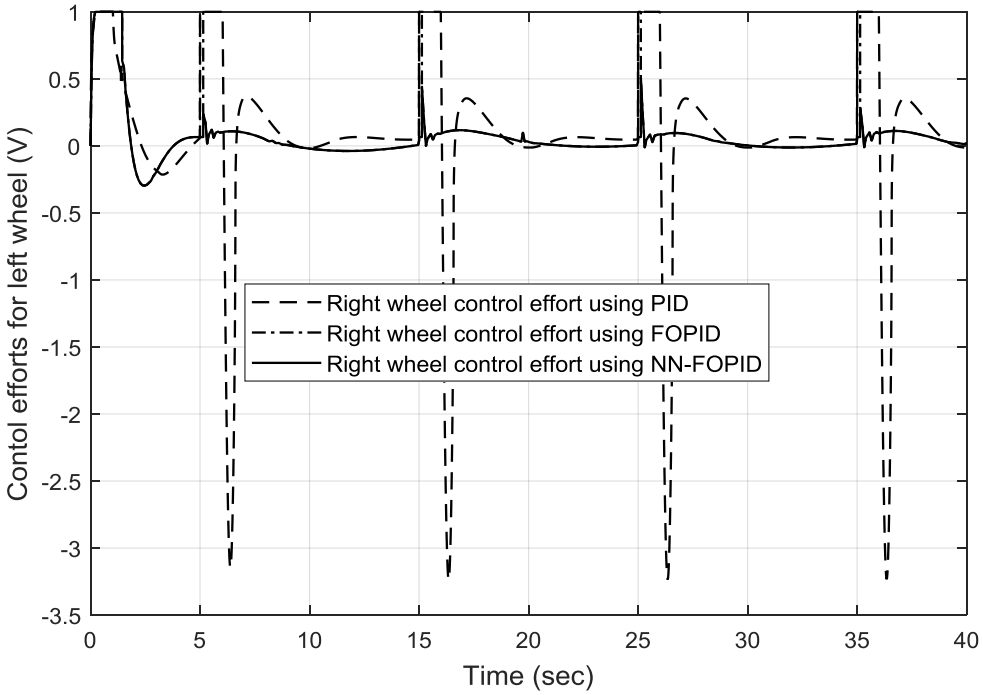



Fig. 5.13 Control efforts for left wheel of square trajectory using PID, FOPID and NN-FOPID controllers.

Figs. 5.14 and 5.15 demonstrate the motion trajectory of the X and Y coordinates respectively. It is quite obvious that the X and Y coordinates have been reduced distinctively. For instance, it can be seen in Fig. 5.16 that the trajectory-tracking error of X coordinate has been eliminated greatly and it reaches the zero level since the UGV commences its movement until the end of the path. In Fig. 5.17, the trajectory tracking error of Y coordinate has been considerably improved and it approaches the zero level.

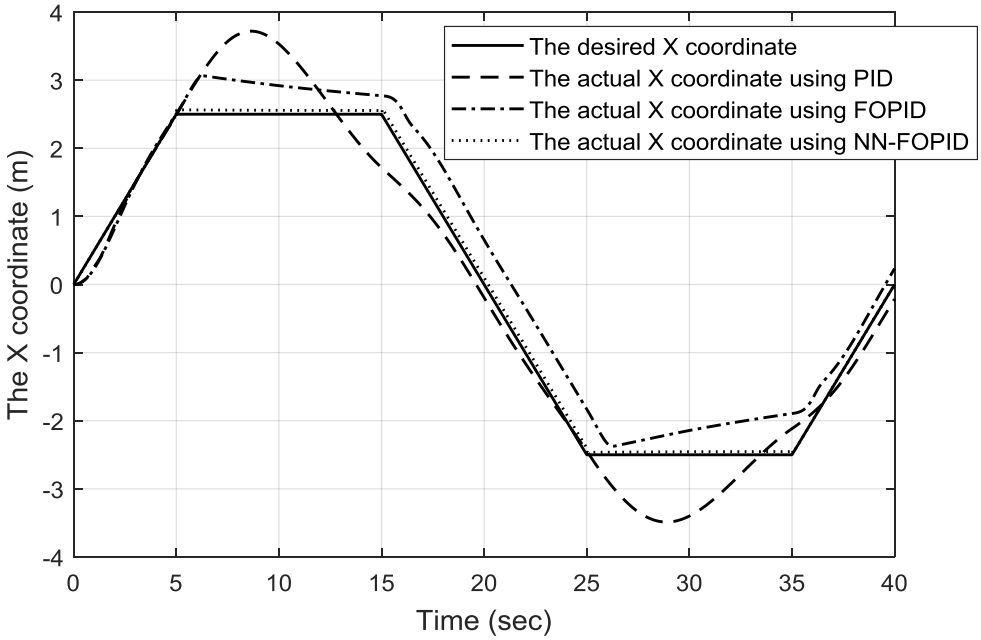



Fig. 5.14 X-coordinates for square trajectory using PID, FOPID and NN-FOPID controllers.

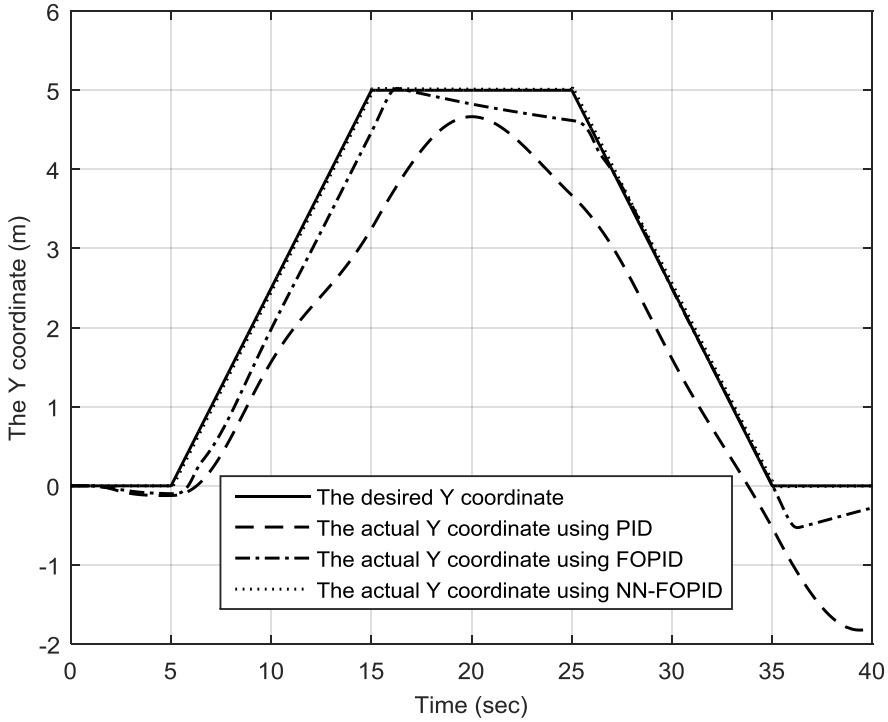



Fig. 5.15 Y-coordinates for square trajectory using PID, FOPID and NN-FOPID controllers.

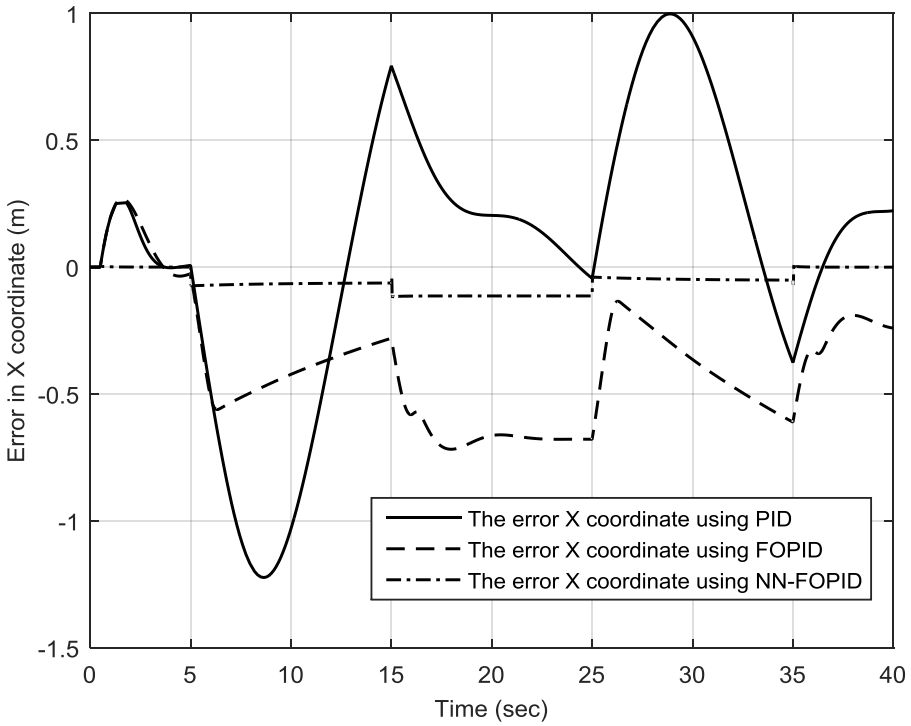



Fig. 5.16 Error in X-coordinates for square trajectory using PID, FOPID and NN-FOPID controllers.

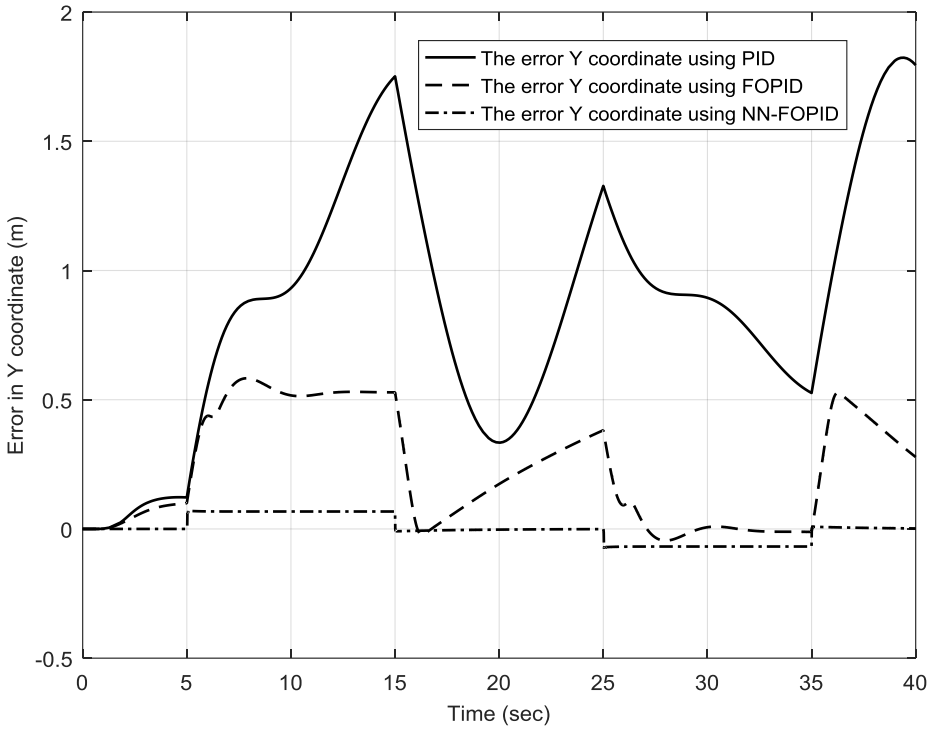



Fig. 5.17 Error in Y-coordinates for square trajectory using PID, FOPID and NN-FOPID controllers.

Finally, the other noticeable difference related to the cost function that is used to optimise the error based on the integral square error is introduced in Fig. 5.18. It is evident there are distinguished improvements of using the neural networks based on the fractional order PID controller comparing to the other methodologies introduced in Chapter 3 and Chapter 4. The tracking control is quite accurate owing to the quick online learning and adaptive capability of the ANNs. The simulation results demonstrate that the adaptive control of the ANN is capable of better tracking performance by dismissing any peculiarities in the behaviour of the UGV whilst tracking the given trajectory.

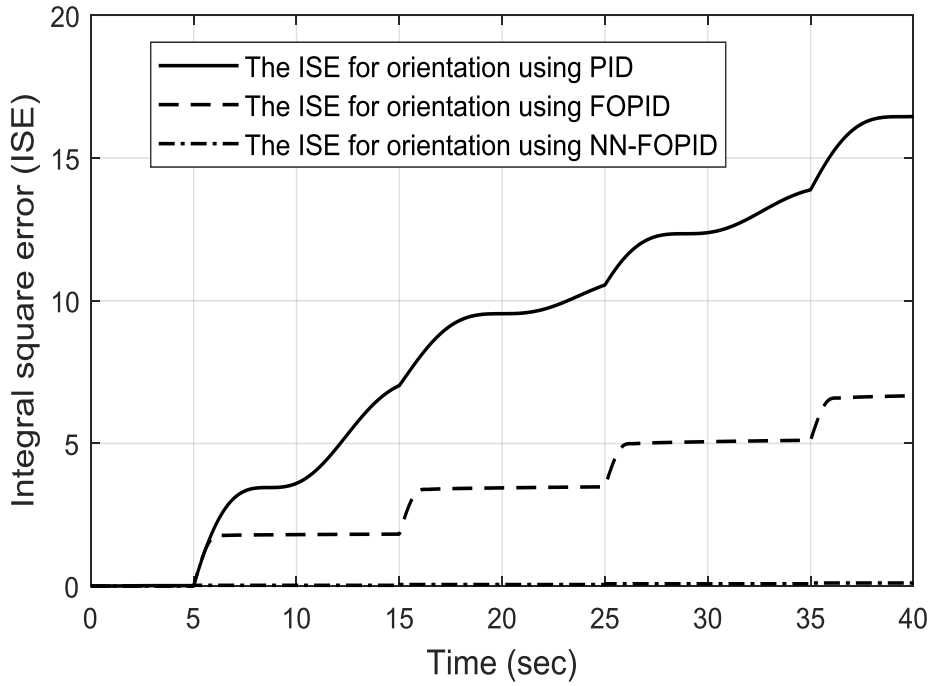



Fig. 5.18 Error in orientation for square trajectory using PID, FOPID and NN-FOPID controllers.

## 5.6 Chapter Summary

In this chapter, two neural networks based on FOPID controllers have been proposed to control the motion of a UGV. The first NN-FOPID controller is used to control the orientation of the UGV whereas the second FOPID-controller is utilised to control the velocity of the UGV. These FOPID-NN controllers have been trained by using the Levenberg–Marquardt algorithm, which have effectively obtained the optimal parameters of the FOPID-NN controllers. The proposed FOPID-NN controllers have demonstrated significant improvements and highly accurate capability to track the square trajectory in comparison with solely FOPID controllers. Additionally, the NN-FOPID-controllers have demonstrated a fast learning capability of tracking the non-continuous gradient square trajectory. The simulation results have confirmed successfully the validation of the introduced NN-FOPID controllers in terms of minimising tracking error and reducing the control efforts, thus, the overall response of the system has been improved. Thereby, from the summary of the main findings, it is observable that the introduced method has a smoother and faster convergence performance of error tracking for orientation angles. Moreover, the simulation results demonstrated the effectiveness of the proposed controller by showing its ability to generate small values of the control input torques for right and left wheels with small sharp spikes.

# Chapter 6

# Navigation of UGV Based on Fuzzy Inference System

## 6.1 Introduction

The increased applications of fuzzy inference systems (FIS) are based on the fact of the simplicity of fuzzy rule-based systems, capable of performing a wide variety tasks without explicit computations and measurements or requirement of prior knowledge about a system mathematical model. These facts make FIS extensively popular among scientists and researchers. FIS provides a means to capture a human mind's expertise. It utilizes heuristic knowledge for representing and accomplishing of a methodology to develop a perceptual action based strategies for UGV's navigation. Furthermore, the methodology of the FIS is strongly helpful in dealing with uncertainties in real world scenarios. Therefore, the importance of the FIS is based on a simple design, an easy implementation and robustness properties. Hence, many real world applications have been using FIS to approach and solve engineering problems. It has been intensively used in facial pattern recognitions, air conditioners, washing machines, vacuum cleaners, and robotic systems.

The applications of the FIS in robotic systems have demonstrated that most existing work has been dedicated to the navigation in static environments, rather than dynamic environments. However, industrial automation applications increasingly demand a development for UGV in dynamic environments based on real world situations. In this chapter, the attempts can be summarised as follows: the FIS is proposed to guide the vehicle in a busy and dynamic environment. A full consideration of the UGV and obstacles with different shapes and sizes has been taken. It is considered a random movement of obstacles in our proposed environment. As a result, this creates a challenge to the operation of the UGV to reach its destination. A new problem statement is proposed for the constructing dynamic environments where multiple obstacle movements are considered randomly with different speeds and directions.

The FIS consists of two fuzzy logic controllers, which they are locally reactive based on UGV's navigation commands. The left and right rear wheels are controlled by the proposed two controllers to achieve an ultimate performance with a minimum power and to navigate safely, by manoeuvring the moving obstacles, all the way towards its destination with minimum delay. The first proposed controller is reacted based on the sensing information of any objects

that are moving near the UGV path or any other obstacles that on the UGV path. The second controller is designed to determine the optimal direction when no obstacles near the UGV platform until the destination point is reached. These two controllers are incorporated into a switching mechanism to choose the operational case that needs to be activated. If the UGV senses an obstacle that is approaching its platform, the first controller will be activated. On the contrary, when the path is clear of obstacles, the second controller will be activated. Thoroughly, both controllers allow the UGV to adjust its trajectory in real-time to avoid collisions with any obstacles in the environment during the navigating and to stop when it reaches its destination. The major advantage of our design is achieving an optimal and a smooth path during navigation. Hence, this will improve the operational performance of the UGV and avoid the random perturbations due to the movement.

## 6.2 Chapter Organisation

The chapter is organised as follows: Section 6.3 present a review of the structure of the fuzzy inference systems. In Section 6.4, the design of our fuzzy inference systems is presented. The environment modelling and the navigation architecture are illustrated in Section 6.5. This describes the implementation process of the obstacle avoidance and target reaching algorithms. Four scenarios are conducted based on experimental simulation results are demonstrated in Section 6.6. In addition, comparisons are provided in Section 6.7 to introduce the benefits of our proposed algorithms and navigation architecture of the state of the art. Finally, Chapter summary is given in Section 6.8.

## 6.3 The Structure of Fuzzy Inference Systems

Fuzzy inference systems are the process of formulating the mapping from inputs to outputs. The inputs and outputs can be defined as the linguistic variables. The mapping, then provides a basis from which decisions can be made based on the internal process of the FIS. The process of the fuzzy inference system involves a variety of terminologies such as fuzzification, inference engine, implication, and aggregation and defuzzification (Lee, 1990; Mamdani and Assilian, 1975).

The architecture of the FIS is demonstrated in Fig. 6.1. FIS can be comprised of a multiple input and output linguistic variables to be controlled. Each linguistic variable has a range of expected values such as ‘0’ to ‘100’ degrees. In addition, there are linguistic terms that

represent categories for the values of a linguistic variable. For example, a linguistic variable 'distance' might include the linguistic terms 'far and near'.

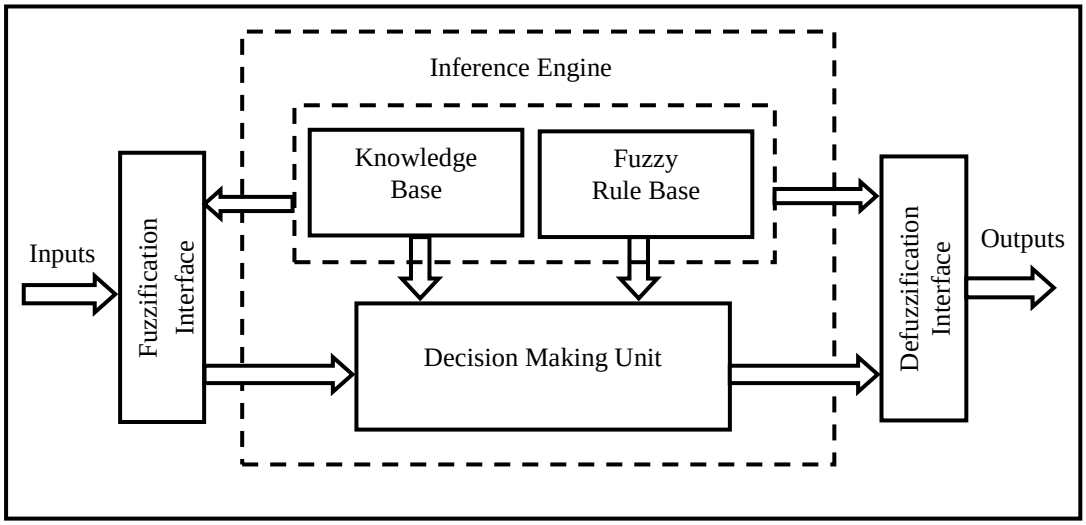



Fig. 6.1 Architecture of fuzzy inference system.

The main components that drive the principle operation of the fuzzy inference system are explained below.

### *6.3.1 Fuzzification*

The composition of the fuzzification is based on numerous of membership functions. The purpose is to map the inputs from a set of sensors or features of those sensors such as amplitude or spectrum to values from '0' to '1' by using a set of input membership functions. The membership functions are numerical functions corresponding to linguistic terms. A membership function represents the degree of membership of linguistic variables within their linguistic terms. The degree of membership is continuous between 0 and 1, where '0' is equal to '0%' of the membership and '1' is equal to '100%' of the membership. The process by which the input values from sensors are scaled and mapped to the domain of fuzzy variables is known as fuzzification. The fuzzy variables are also known as linguistic variables that are determined based on intuition (from knowledge) or inference (known facts). These linguistic variables can be either continuous or discrete theoretically. However, practically, it should be discrete. Fuzzification can be classified as a two-step process: Assign fuzzy labels and Assign numerical meaning to each label (Sumathi and Paneerselvam, 2010).

**1) Assign fuzzy label:**

The real inputs of the FIS are called crisp inputs. Each crisp input is assigned a fuzzy label in a universe of discourse. For example, for the input parameter, sensor distance, fuzzy labels

can be “far” and “near”. In addition, every crisp input can be assigned multiple labels. As the number of labels increases the resolution of the process is better. In some cases, assigning large number of labels leads to a large computational time and this might make a fuzzy system unstable.

**2) Assign numerical meaning:**

Membership functions are formed to assign a numerical meaning to each label. The range of the input value that corresponds to a specific label can be identified by the related membership function. Although there are different shapes of membership functions, triangular and trapezoidal membership functions are commonly used to avoid time and space complexity. The crisp inputs of FIS are firstly fuzzified into linguistic values before the inference engine proceeds in processing with the rule base.

### *6.3.2 Inference Engine*

The inference engine is a major unit that represents a fuzzy knowledge base and a decision making fuzzy inference system. The FIS formulates suitable rules and based upon the rules decisions are made. This is mainly based on the concepts of a set of fuzzy using statements of “IF-THEN”. FIS uses the “IF-THEN” statements and connectors that would be presented in rule statements such as “OR” or “AND”. The inputs of the FIS can take either fuzzy inputs or crisp inputs. However, the FIS produces outputs as fuzzy sets. When the FIS is used as a controller, it is necessary to have a crisp output. Therefore, a defuzzification method is adopted to extract crisp values that represent the best fuzzy set (Driankov and Saffiotti, 2001).

After the aggregation of fuzzy rules, a fuzzy set for each output variable needs defuzzification. It is possible to use a single spike as the output membership function rather than a distributed fuzzy set. This is known as a singleton output membership function, and it can be thought of as a pre-defuzzified fuzzy set. It enhances the efficiency of the defuzzification process because it greatly simplifies the computation required by Mamdani method (Mamdani, 1977). It finds the centroid of a two-dimensional function rather than integrating across ta two-dimensional function to find the centroid and the weighted average.

### *6.3.3 Generation of Fuzzy Rules*

Fuzzy rules describe the relationships between input and output linguistic variables based on their linguistic terms. For example, a rule might be defined such as: IF left distance is near AND right distance is near, THEN speed setting is minimum. The clauses “front distance is

near" and "right distance is near" are the antecedents of this rule. The AND connective specifies how a FIS relates two antecedents to determine a truth-value for an aggregated rule antecedent. The clause "speed setting is minimum" is the consequent of a rule. The fuzzy rules are formed by integrating linguistic variables using assignment conditional based on a set of IF-THEN fuzzy rules. These rules are generated based on conditional statements. Examples of rules formation are given as follows:

IF $antecedent_1$ AND $antecedent_2$ AND…AND $antecedent_n$ THEN *consequent*

IF $antecedent_1$ OR $antecedent_2$ OR…OR $antecedent_n$ THEN *consequent*

### *6.3.4 Aggregation of Fuzzy Rules*

The rule based system constructs of several rules and each rule provides an output or a consequent. The consequent part also known as conclusion and it is a unique for every individual rule that has been executed based on input parameters. An overall conclusion has to be obtained from individual consequents. The method of obtaining the overall conclusion from the set of rules is called an aggregation of rules. Using 'AND' or 'OR' operators, a final decision is made on the output of the fuzzy set.

Fuzzy rules can be aggregated by using the 'AND' or 'OR' connectives. The process of aggregating the rules using 'AND' connective is known as conjunctive aggregation and the process of aggregating the rules using 'OR' connective is known as disjunctive aggregation. Examples of a conjunctive and disjunctive aggregation are given below:

1) **Conjunctive aggregation:**

Consequent = $Consequent_1$ AND $Consequent_2$ AND ... AND $Consequent_n$

2) **Disjunctive aggregation:**

Consequent = $Consequent_1$ OR $Consequent_2$ OR ... OR $Consequent_n$

### *6.3.5 Defuzzification*

The necessity of converting fuzzy quantities into crisp quantities is to obtain real values for further processing in related applications. Hence, the defuzziffication is the process of converting the degrees of membership of output linguistic variables into numerical values. The transformation of fuzzy information into crisp outputs is necessary in order to acquire the required control signals. The control signals can be applied to control a plant when a fuzzy system is used as a controller. There are several defuzzification methods, in which the commonly used is the centre of area (CoA) (Branson and Lilly, 2001). The CoA method takes the output distribution and finds its centre of mass to come up with one crisp number.

In the CoA method, the FIS calculates the area under the scaled membership functions and within the range of the output variable. Thus, it uses the following equation to calculate the geometric centre of this area.

$$CoA = \frac{\int_{x_{min}}^{x_{max}} f(x).xdx}{\int_{x_{min}}^{x_{max}} f(x)dx} \tag{4.1}$$

where,

CoA is the centre of area,

x is the value of the linguistic variable, and

$x_{min}$ and $x_{max}$ represent the range of the linguistic variable.

Fig. 6.2 illustrates the CoA defuzzification method. In this figure, μ is the degree of membership, and the shaded portion of the graph represents the area under the scaled membership functions.

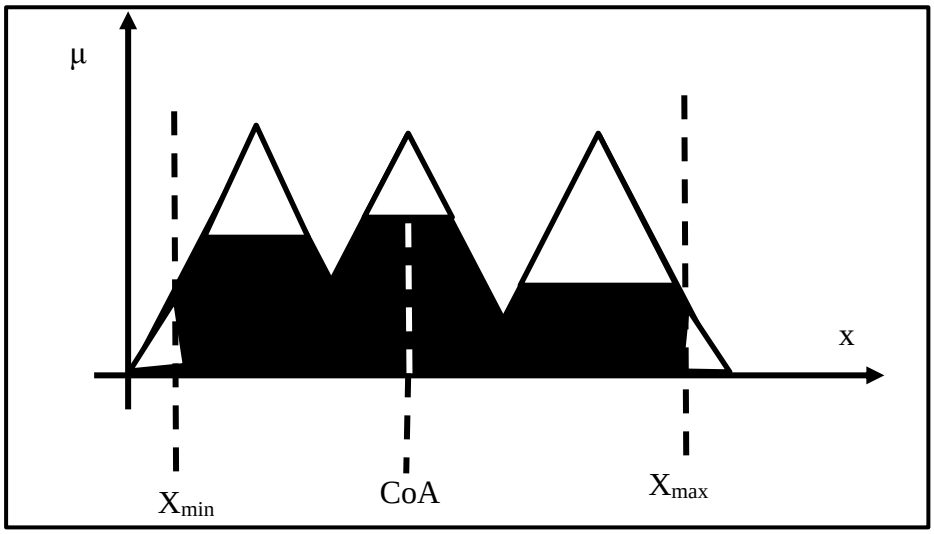



Fig. 6.2 Centre of area method.

## 6.4 Design of FIS for Navigation

In this work, the proposed intelligent controller is implemented based on the fuzzy inference system. The FIS consists of two fuzzy logic controllers as demonstrated in Fig. 6.3 that are reactive based upon the navigation of the unmanned ground vehicle. The wheels on the left and right sides are controlled by those FIS controllers for achieving the objective of the UGV to navigate safely and reach the target point without colliding moving obstacles appeared on its path. The first FIS controller is reacted based upon sensing obstacles near to the vehicle's platform such as left distance (LD), front distance (FD) and right distance (RD). The second FIS controller responds for choosing the optimal direction in case of non-obstacles surrounding the vehicle's platform based on an angle difference (AD) between the vehicle's heading and

the target's orientation. Therefore, the UGV will be able of adjusting its online trajectory to avoid collision with obstacles in the workspace through approaching a given target point.

The working of FIS can be explained as follows. The crisp inputs, i.e. sensory information and angle difference are converted into fuzzy sets by using the fuzzification method based on Mamdani type. Then, the rules are formed based on the data and knowledge bases which are jointly referred to make decisions. Defuzzification is used to convert fuzzy values to the real world values which they represent the final outputs that drive the wheels of the UGV.

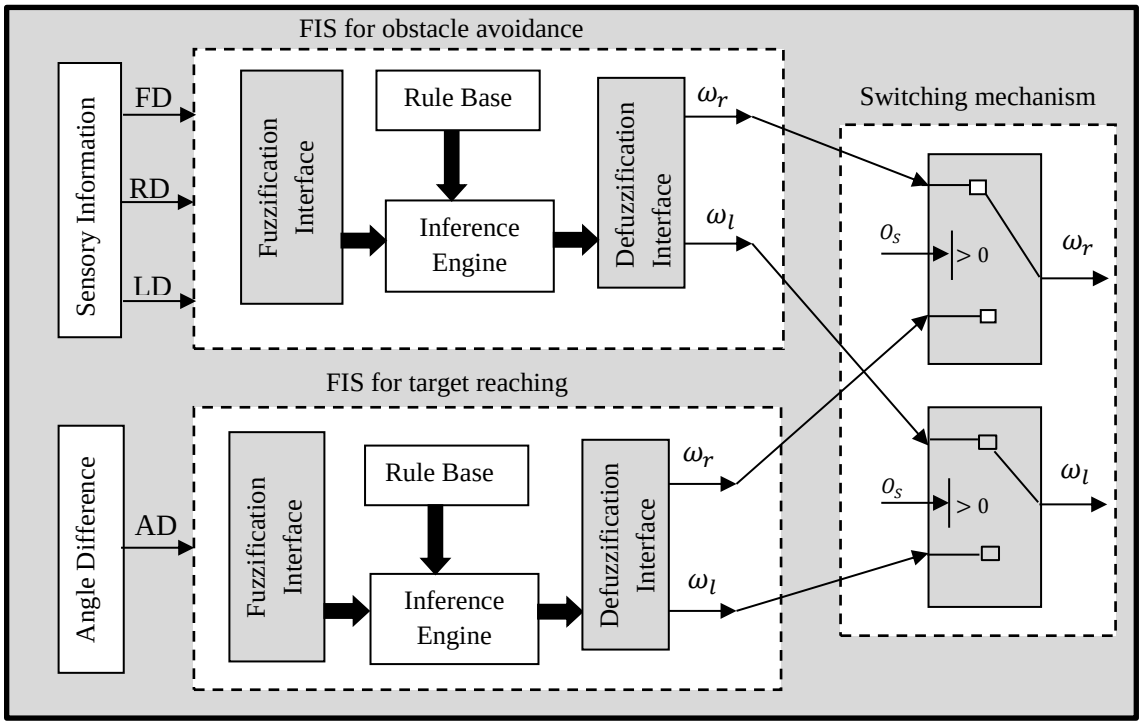



Fig. 6.3 Block diagram of proposed two FIS controllers.

The obstacle avoidance fuzzy inference system (OA-FIS) controller is designed to avoid collision with obstacles by using the sensory data. The target reaching fuzzy inference system (TR-FIS) controller aims to make the decision for choosing the optimal direction. Two switches are used to combine the output from the two FIS controllers. The output signal from the first switch is utilised to drive the motor of the right wheel whereas the motor of the left wheel is driven by the signal from the second switch.

The operation of the FIS controllers can be summarised as follows: when there is no obstacle on the UGV path, the TR-FIS controller will be activated to determine the best route for the UGV to take and reach its destination. Whereas, if there is an obstacle on the path of the UGV, the OA-FIS controller will be activated to enable an obstacle avoidance strategy. The

switching mechanism between those two controllers will be decided based on information provided by a signal called obstacle sensing (OS). This OS signal indicates '0' when there is no obstacles and '1' when the UGV sensors detect an obstructing obstacle approaching the UGV's path. The outputs of the switching block will be the angular velocities of the right and left wheels of the UGV. These velocities are then fed into the vehicle's model in order to obtain the instantaneous UGV coordinates.

### *6.4.1 Fuzzy Logic Controller for Obstacle Avoidance*

The main objective of this controller is to ensure that the UGV is capable of avoiding collision when moving obstacles that are diffused in its path. This controller is constructed based on the sensing information that received from the UGV fitted sensors. The sensory information is collected from three sensors. These sensors are placed on the forefront UGV in three different positions to estimate the left distance, front distance and right distance of the UGV with respect to any obstacle as shown in Fig. 6.4. The three estimated distances are supplied as inputs to the obstacle avoidance controller. Then, the obstacle controller produces the angular velocities for the left and right wheels of the UGV that are symbolised as $\omega_l$ for left wheel angular velocity and $\omega_r$ for the right wheel angular velocity. Thus, this controller actually regulates the direction of the UGV motion based on the obstacles locations by changing the angular velocities for each of the rear wheels.

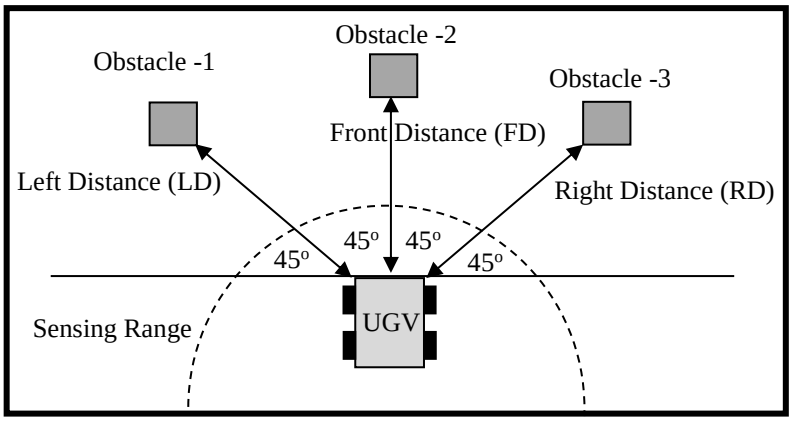



Fig. 6.4 Schematic diagram for sensory information.

The fuzzy system is implemented with two trapezoidal membership functions for each input variable of three distances, LD, RD and FD. The input variables for the obstacle avoidance controller are defined as the distance from the left (LD), distance from the right (RD) and

distance from the front (FD). The estimated measurements are categorised as either near or far. A fuzzy logic controller for obstacle avoidance is designed using the Mamdani fuzzy inference mechanism. The knowledge base of the system consists of multiple rules that are given in Table 6.1. It shows that the output values of LD, RD and FD are combined and translated to generate commands to the motors of the individual wheels independently based on the following specified angular velocities of $\omega_l$ and $\omega_r$: Backward (BF), Backward Slow (BS), Forward Slow (FS) and Forward (FF). That means each output has five triangular membership functions. The defuzzification process is computed based on the centroid defuzzification technique, which is also called the centre of gravity.

Table 6.1 Rules base of the OA controller.

| **Rule No.** | **Inputs** | | | **Outputs** | |
|---|---|---|---|---|---|
| | **Front Distance (FD)** | **Right Distance (RD)** | **Left Distance (LD)** | **Right angular velocity ($\omega_r$)** | **Right angular velocity ($\omega_l$)** |
| 1 | Near | Near | Near | BF | BF |
| 2 | Near | Near | Far | FS | BS |
| 3 | Near | Far | Near | BF | FS |
| 4 | Far | Near | Near | FS | FS |
| 5 | Near | Far | Far | BS | FF |
| 6 | Far | Near | Far | FS | BS |
| 7 | Far | Far | Near | BS | FS |
| 8 | Far | Far | Far | FF | FF |

Fig. 6.5 demonstrates the membership functions of the OA controller's inputs, which are the sensory information of the left, right and front distances. For example, the linguistic variable 'Near Distance' has full membership (100%) within the linguistic term far at all far locations between 40 and 100 degrees, no membership (0%) within that term at 20 degrees or less, and partial membership at all between '20' and '40' degrees. Fig. 6.6 shows the membership functions for the outputs of the OA controller, which are the angular velocity of the rear right driving wheel and the angular velocity of the rear left driving wheel of the UGV. The surfaces of the three inputs FIS named (FD, RD, and LD) to its first output (right angular velocity) are depicted in Figs. 6.7(a), (b) and (c). Likewise, the same surfaces views can be obtained regarding the second output, which represents the left angular velocity. The surface viewer simply shows the mapping graphically between any two inputs and an output.

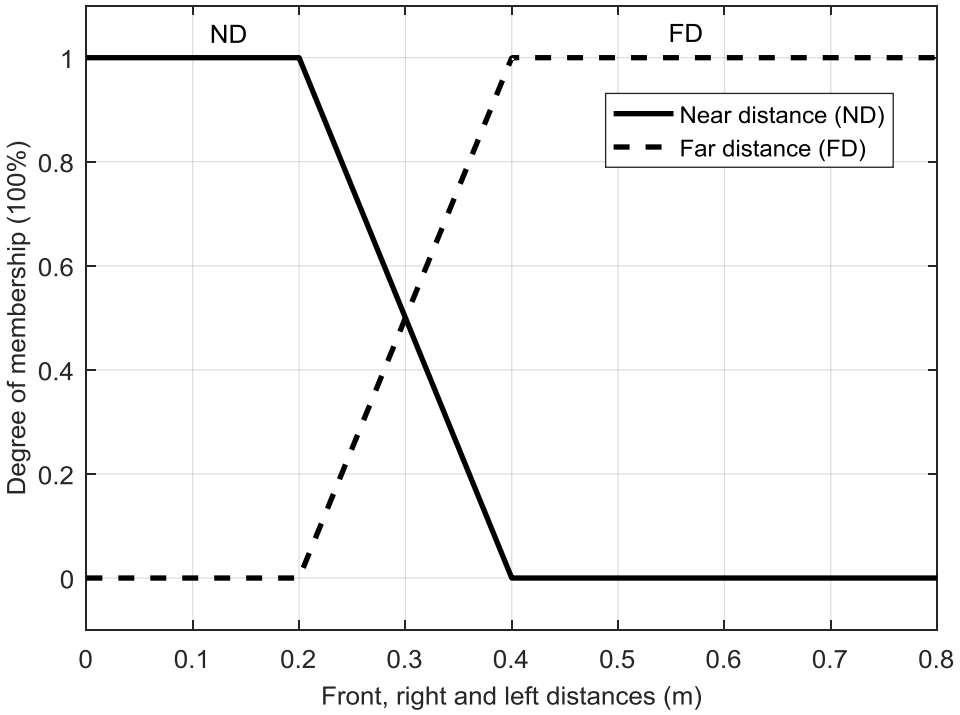



Fig. 6.5 Membership functions of right and left angular velocities.

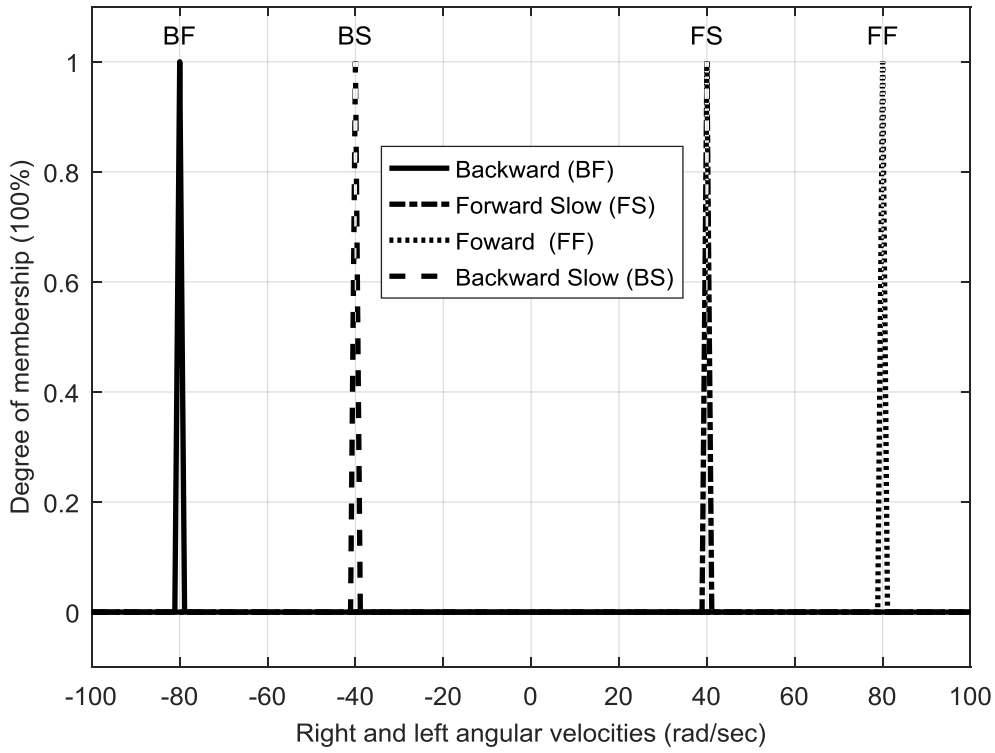



Fig. 6.6 Membership functions of front, right and left distances.

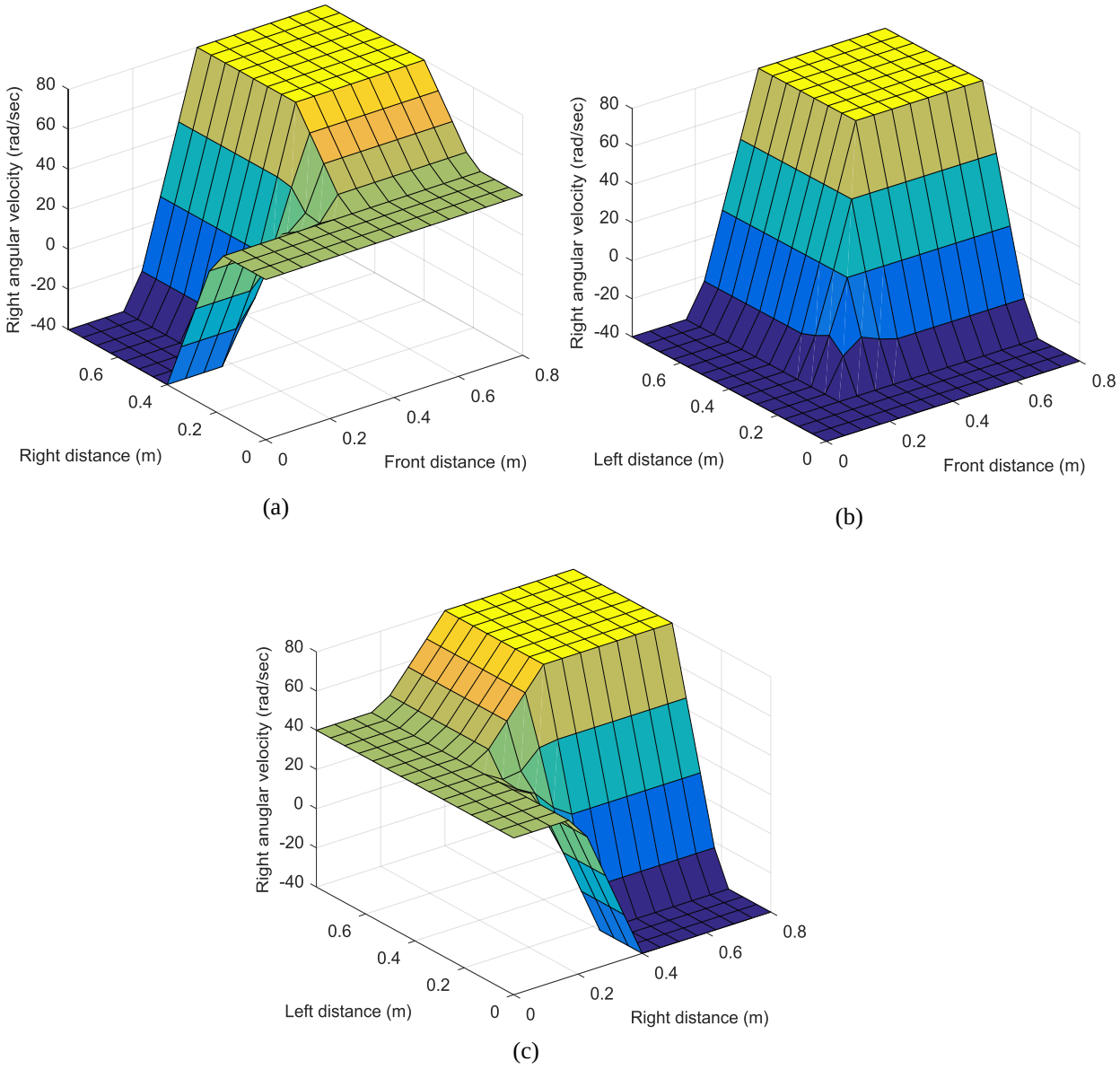



Fig. 6.7 Surface viewer for right angular velocity against a) right and front distances; b) front and left distances; c) left and right distances.

### *6.4.2 Fuzzy Logic Controller for Target Reaching*

The target-reaching controller is designed to enable the UGV to reach its target destination in the shortest distance. In this section, the FIS controller is implemented using the same principle utilised for the obstacle avoidance controller. The input of this controller is an angle which represents the difference between the heading of the UGV and the targeted angle. This angle difference (AD) is computed as shown in Fig. 6.8. The outputs of this controller are obtained using the same technique that used in obtaining the FIS-OA controller outputs. The

input variable of the target reaching controller is the angle difference (AD) which can be big negative (BN), small negative (SN), zero (Z), small positive (SP) or big positive (BP).

The output variables are the angular velocities for the rear driving wheels ($\omega_l$ and $\omega_r$) which are categorised as Backward (BF), Backward Slow (BS), Forward Slow (FS) or Forward (FF). Fig. 6.9 represents the membership functions of the angle difference (AD) input. The membership functions of the target reaching (TR) controller outputs, which are the left and right angular velocities, are same as in Fig. 6.6 given preciously. The rules of the target reaching controller are given in the Table 6.2. The relationships of the surface view between the angle difference (AD) variable and the left and right angular velocities, the outputs of the target-reaching controller, are illustrated in Fig. 6.10. The surface viewer is shown in a two-dimensional graph because it only represents the relationship between one input and one output.

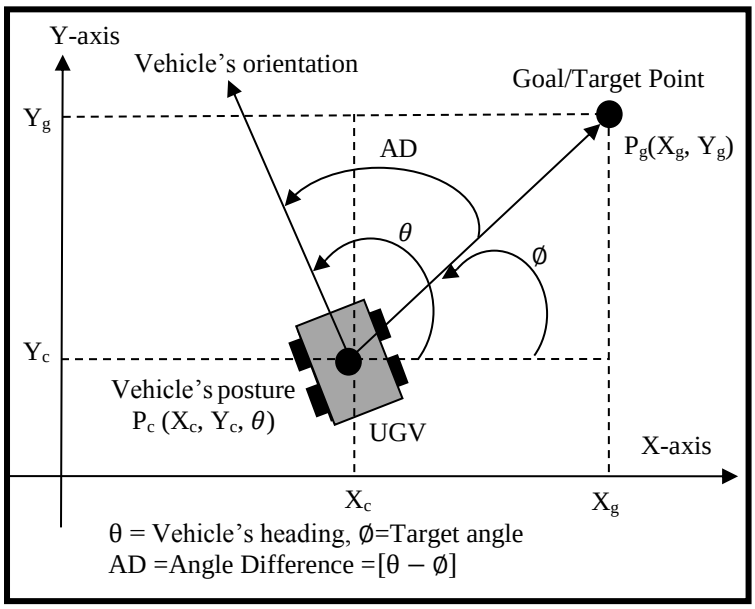



Fig. 6.8 Coordinates of instantaneous localisation in a workspace.

Table 6.2 Rules base of the TR controller.

| **Rule No.** | **Input** | **Outputs** | |
|---|---|---|---|
| | **Angle Difference (AD)** | **Right angular velocity ($\omega_r$)** | **Right angular velocity ($\omega_l$)** |
| 1 | SP | BS | FS |
| 2 | BP | BS | FS |
| 3 | SN | FS | BS |
| 4 | BN | FS | BS |
| 5 | Zero | FF | FF |

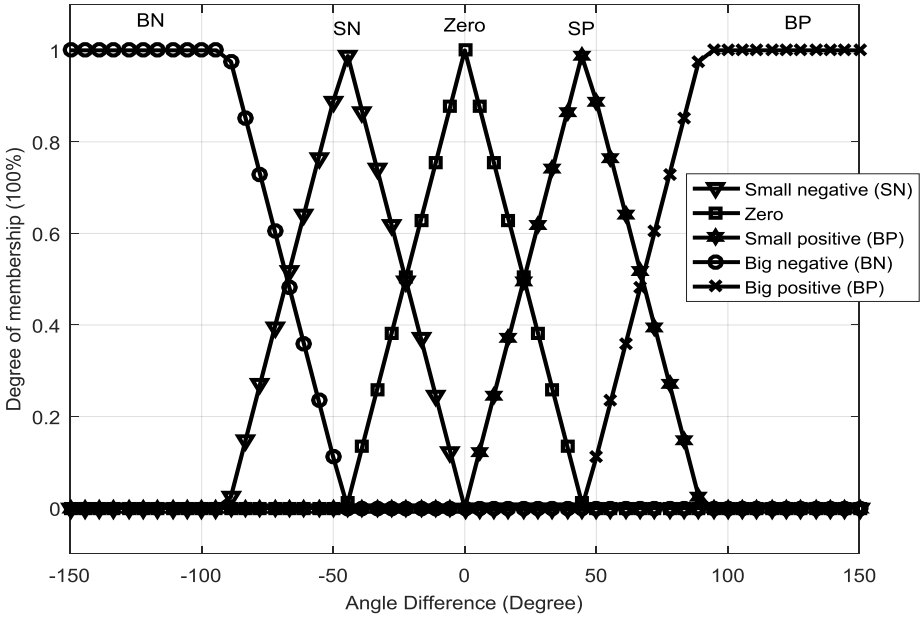



Fig. 6.9 Membership functions of angle difference.

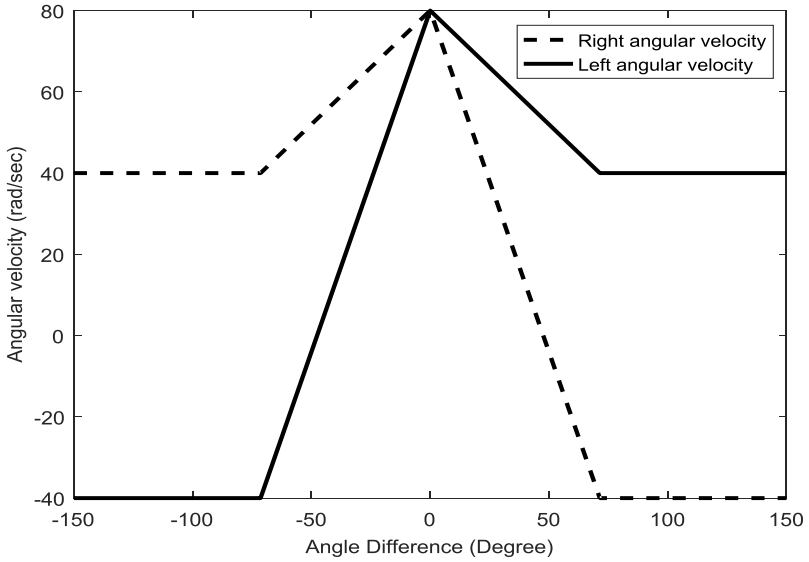



Fig. 6.10 Surface viewer for angle difference.

## 6.5 Navigation Architecture and Environment Modelling

This section deals with the implementation of the UGV navigation platform and the workspace. The objective of the navigation architecture and environment modelling is to show how to develop and perform the navigation of the UGV based on the implemented platform and the workspace environment. The two fuzzy logic controllers are integrated through a switching mechanism as discussed in the previous section. The outputs of the switching block drive the motors of the driving wheels and move the UGV. When the angular velocities (the

outputs) are identical, it means that the UGV will move in a straight line, but if the angular velocities are different, the UGV will be in steering situation either to the right or the left based on the obstacles locations. The dynamic environment of the workspace where the UGV navigates is implemented. The dynamic environment is based on dynamic obstacles in addition to the UGV's platform. The Block diagram of the navigation architecture is shown in Fig. 6.11. As demonstrated, the environment modelling has five inputs and five outputs. The inputs are classified into two groups which are the actual position of the UGV (three inputs) $P_c = (X_c, Y_c, \theta)$ and the target point coordinate (two inputs).

The outputs of the environment modelling, which are the inputs of the controllers and switching boxes, are categorised into three groups. First, the sensory information includes: front distance (FD), right distance (RD) and left distance (LD). This sensory information provides the OA-FIS controller with the necessary information to obtain the accurate angular velocities for the driving wheels for manoeuvring and avoid collisions. Second, the angle difference (AD) which represents the resultant direction of the UGV towards the destination. This angle is connected to the TR-FIS controller as an input to obtain the angular velocities for the rear driving wheels so that the UGV can move in a straight line towards its destination when there is no obstacle in its path. Third, the obstacle sensing (OS) is a switching mode to activate one of two FIS controllers as required, depending on the UGV surroundings. The S-function manipulates the interconnection between the inputs and the outputs. It has been written by suing MATLAB coding for creating a simulation platform for the unmanned ground vehicle navigation and the surrounding environment. In summary, the main parameters of the navigation platform are stated as follows:

**Remark 1.** The front, right and left distances represent the shortest distances between the vehicle and obstacles. The sensory information is modelled by assuming these three sensors are placed on a vehicle's platform, and each senor carries the information for three directions of the platform. These sensor outputs change depending on the distance between the instantaneous positions of the vehicle.

**Remark 2.** The angle difference represents the difference between the vehicle's heading and the target point.

**Remark 3.** Obstacle sensing (OS) signal is generated in accordance to the measured distances (front, right, left) from the sensory information. If the vehicle does not sense any obstacles in its path, this OS parameter will indicate '0', and '1' if the vehicle senses any obstacles near to its platform.

**Remark 4.** The clock timer is used for measuring the simulation running time that the vehicle elapsed to reach the destination in the platform.

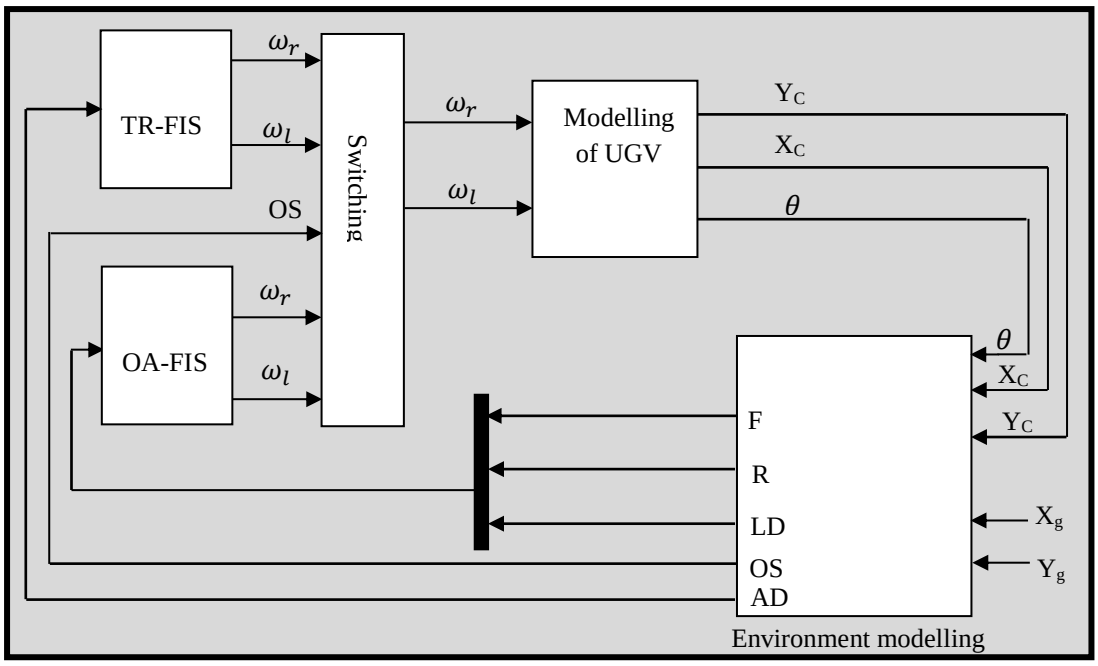



Fig. 6.11 Block diagram of proposed FIS for UGV.

The composition of the navigation platform constitutes of a multi-controller architecture. Hence, a hierarchy switching mechanism is provided to select the appropriate controller based on a specific scenario. The Algorithm 6.1 illustrates the logic of the hierarchical selection of the switching mechanism. The purpose of the algorithm is to ascertain one controller is always activated as needed. For instance, the hierarchical switching mechanism activates the obstacle avoidance controller when an obstructing obstacle encounters the UGV's movement. On the other hand, the stimulus of the target reaching controller occurs when the path of UGV is free of obstacles by a trigger of the obstacle sensing parameter.

**Algorithm 6.1:** Hierarchy of switching mechanism

**Inputs:** All distance parameters of the closed obstacles in three directions (left, front and right); the localisation of the initial point $P_o$ ($X_o$, $Y_o$).and goal point $P_g$ ($X_g$, $Y_g$).

**Outputs:** Define which controller to be activated.

```
1 if It exists at least on obstructing obstacle then
2 |     Activate fuzzy logic controller for obstacle avoidance (Algorithm 6.2)
3 else
4 |     Activate fuzzy logic controller for target reaching (Algorithm 6.3)
5 end
```

The algorithm 6.2 describes how the obstacle avoidance reacts when obstructing obstacles are hindering the movement of the UGV towards a particular destination. It also demonstrates a sequence of events that happen in reaction to a specific case at a time. The Algorithm 6.3 demonstrates the target reaching response when there are no obstructing obstacles that are facing the UGV's movement. Hence, the UGV will act according the angle between the current orientation of the UGV and the heading of its destination as explained. It is notably that the target reaching is performed when the UGV terminates at the arrival position.

**Algorithm 6.2:** Obstacle Detection

**Inputs:** All distance parameters of the obstructing obstacles in three directions (left, front and right), the localisation of the goal point $P_g$ ($X_g$, $Y_g$).

**Outputs:** The index parameter 'OS' when obstructing obstacle is detected, driving angular velocities ($\omega_r$, $\omega_l$) for avoiding obstructing obstacles.

```
for Each Sensor (Left, front, and right), measure distances of each obstacle.
    If The obstacle is within the threshold range then
        Add the obstacle to the obstructing obstacles list
    end
end
if obstructing obstacles list ≠ zero then
    identify the location of the obstacle (left, front right)
    If The detected obstacle is on the left then
        the UGV turns right
    end
    If The detected obstacle is on the right then
        the UGV turns left
    end
    If The detected obstacle is on the front then
        the UGV turns either left or right based on the safest way
    end
    If The more than one obstructing obstacle is detected then
        the UGV will seek another obstacle free path, it will stop when it is stuck.
    end
else
    No obstructing obstacle is detected; switch to Algorithm 6.3.
end
```

**Algorithm 6.3:** Target Reaching

**Inputs:** All distance parameters of the closed obstacles in three directions (left, front and right), the localisation of the goal point $P_g$ ($X_g$, $Y_g$) and the actual localisation of the UGV i.e. $P_c$ ($X_c$, $Y_c$, $\theta$), current position of the left and right wheel, and L is the distance between driving wheels

**Outputs:** The predicted position of the UGV at a time, $P_t' = f(x, y, \theta, \Delta S_r, \Delta S_l)$, driving angular velocities ($\omega_r$, $\omega_l$) for target reaching.

**for** j=1:1:max(size($X_g$))
  **for** i=1:1:max(size($Y_g$))
    **If** $|X_g - X_c|=0$ && $|Y_g - Y_c|=0$ **then**
      The target is reached and stop the UGV.
    **elseif** the distance sensors detect obstructing obstacles **then**
      Switch to Algorithm 6.2.
    **else**
      Determine the new position of the UGV at a time t, can be computed from the previous estimate $x_{t-1}$ and the odometric integration of the movement. The motion is based on the equations (3.11-3.14) given in section 3.4.1 Kinematic Modelling in Chapter 3.
      $x_c = S\cos(\theta)$
      $y_c = S\sin(\theta)$
      $\theta = \frac{S_r - S_l}{L}$
      $S = \frac{S_r + S_l}{2}$
      Updated position is, $P_t' = f(x, y, \theta, S_r, S_l) = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{S_r+S_l}{2}\cos(\theta_{t-1} + \frac{S_r-S_l}{L}) \\ \frac{S_r+S_l}{2}\cos(\theta_{t-1} + \frac{S_r-S_l}{L}) \\ \frac{S_r-S_l}{L} \end{bmatrix}$
      The orientation angle is, $\emptyset = \tan^{-1}([X_c \quad Y_c], [X_g \quad Y_g])$
      The angel difference is, $AD = [\theta - \emptyset]$
    **end**
  **end**
**end**

## 6.6 Simulation Results

The unmanned ground vehicle platform has been simulated using MATLAB-Simulink software package to mimic and verify the effectiveness of the controllers based on the fuzzy inference system. In the simulation environments, different scenarios have been established to validate the operational performance. Each scenario comprises multiple moving obstacles, they are randomly placed based in their topology. The obstacles are considered to be in different sizes and might move at various velocities. The initial and destination points of the unmanned ground vehicle are also randomly chosen at diverse positions to generate a feasible path. To investigate the performance of the two FIS controllers, four scenarios are presented, each of which demonstrates different performance of path planning based on dissimilar sizes, velocities and orientations of obstacles.

### *6.6.1 Scenario-I: Dynamic obstacles with similar sizes and velocities*

In this scenario, in addition to the UGV, there are six moving objects in the environment. The path generation is simulated based on such a workspace construction. The six objects are moving at a constant velocity of 2.42 m/s. As a reference, the coordinates of the starting point is marked as the origin $Po\ (0, 0)$ and the coordinates of the destination is $Pg\ (15, 15)$. Fig. 6.12 demonstrates that the UGV has successfully manoeuvred the obstacles by changing its moving direction and avoided collisions. The target destination is reached by the UGV, which means the TR-FIS controller changed the direction of the UGV movement back to the destination after passing each obstacle. As discussed earlier, the decisions to avoid collisions with obstacles are made by the OA-FIS controller according to the distances between the left, right and the front of the UGV and the obstacles. The journey of the UGV from the starting point to the target destination in this scenario has elapsed 3.21 seconds. Based on the constructed workspace in this scenario, the motion of the UGV reveals that the generated path is feasible and smooth during completing the navigation task.

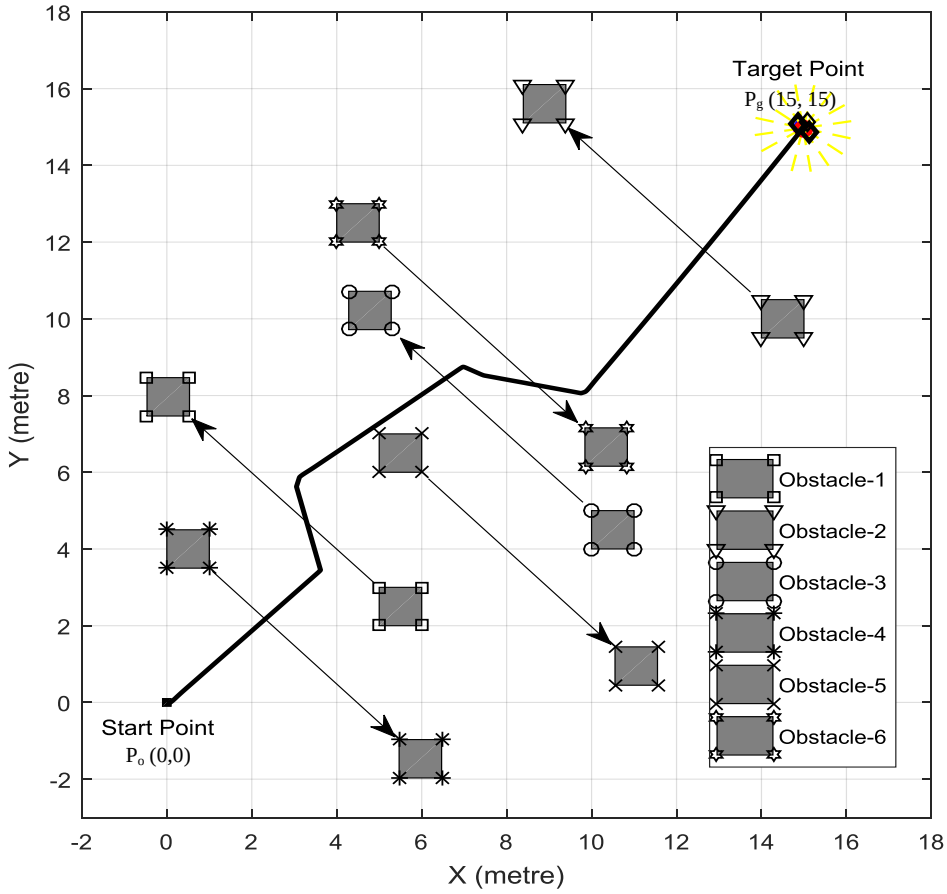



Fig. 6.12 Navigation platform and topology for scenario-I using FIS.

The orientation of the UGV is shown in Fig. 6.13. It demonstrates the behaviour of the path generation with respect to the changing in directions. It obvious that the first changing has occurred when obstacle no.1 has approached to the UGV in the first period. Hence, when the UGV has passed this obstacle, it has amended the heading toward the required destination. However, when obstacle no.6 confronts the UGV's path, the UGV also has orientated into a different direction that is clear of obstacles. It is noticeable that there are two changes appearing on the second period. In fact, as the obstacle is still moving, thus, the UGV has carried on its motion until it has completely passed this obstacle, then; it heads to the given destination. Fig. 6.14 demonstrates the linear velocity response of the UGV when it moves and responds to the changes in the workspace repeatedly. The velocity profile illustrates that the speed has declined rapidly from its peak value and reach the zero, this has happened expectedly when the turning occurs. Hence, the UGV should be at the minimum speed to make a feasible steering.

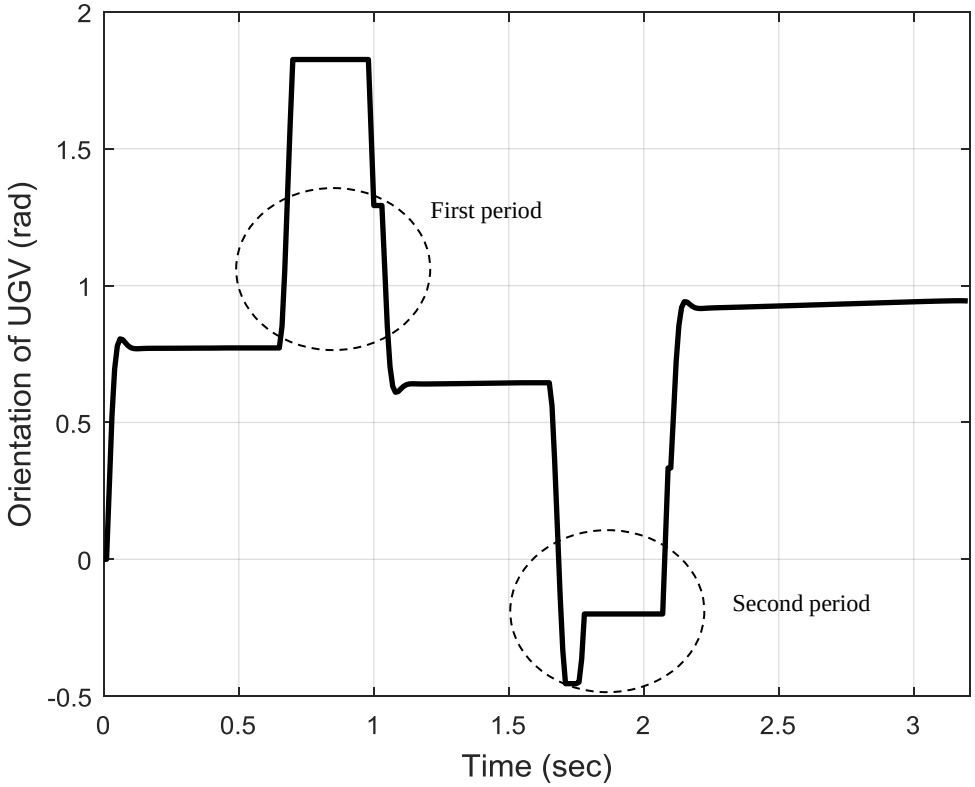



Fig. 6.13 Orientation of UGV in scenario-I using FIS.

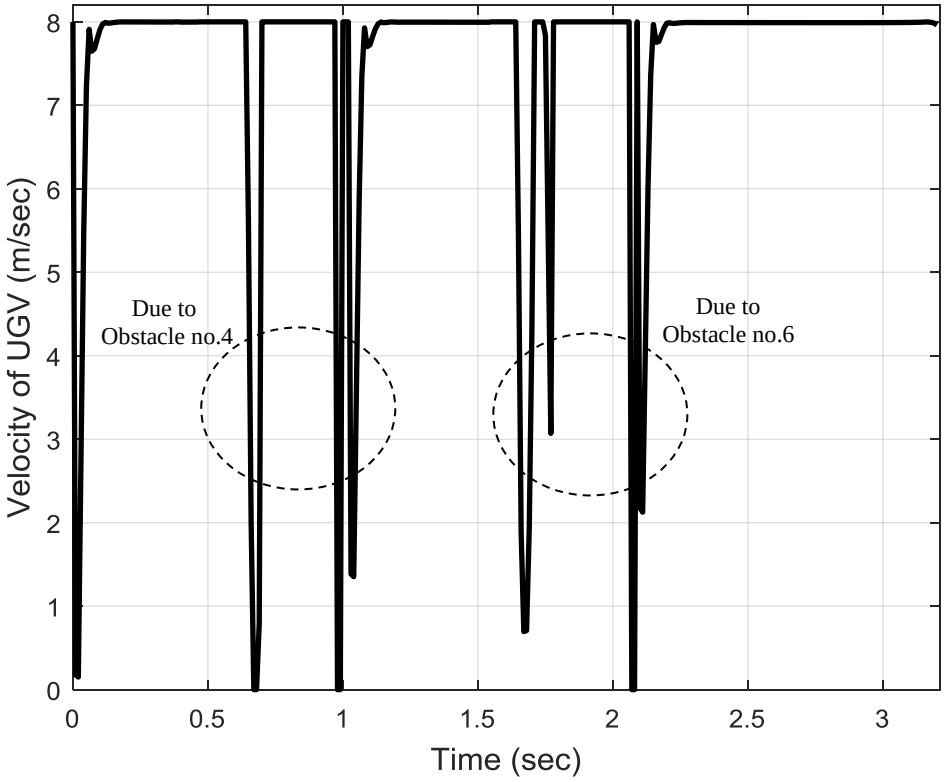



Fig. 6.14 Linear velocity for UGV in scenario-I using FIS.

The two FIS controllers can be analysed and discussed in more detail for understanding the system reaction. The actual path of the UGV is generated based on the individual responses of controller designs. The angular velocity of the FIS controller for the target reaching which is shown in Fig. 6.15, it introduces the behaviour of the right and left wheels to generate the actual path based on the steering wheel angle. This result demonstrates that both of the right and left wheel move at a constant speed unless it faces obstacles. For instance, it is observable that two changes have occurred in two periods in response to obstacles no.4 and no.6. The main reported observations are that; in the first period, the right angular velocity is higher than the left angular velocity. Hence, the orientation has changed to the left. In contrast, in the second period, the left angular velocity is higher than the right, which brings the UGV to the right as noticed on navigation platform of the first scenario. Likewise, it is noticeable that in Fig. 6.16, the FIS of obstacle avoidance has responded at the same period due to the approaching of obstacles. Subsequently, the two FIS controllers are combined into a switching mechanism. Hence, the aggregation of the left and right angular velocity has produced the total angular velocity of the UGV as shown in Fig. 6.17.

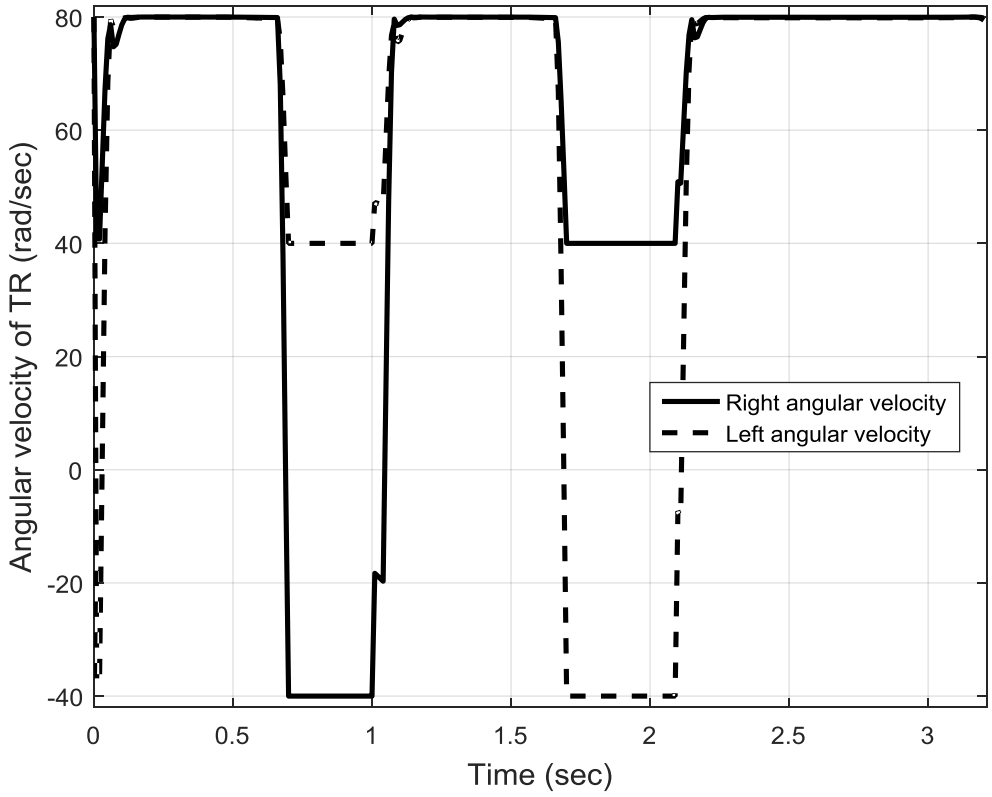



Fig. 6.15 Angular velocity for target reaching of FIS in scenario-I.

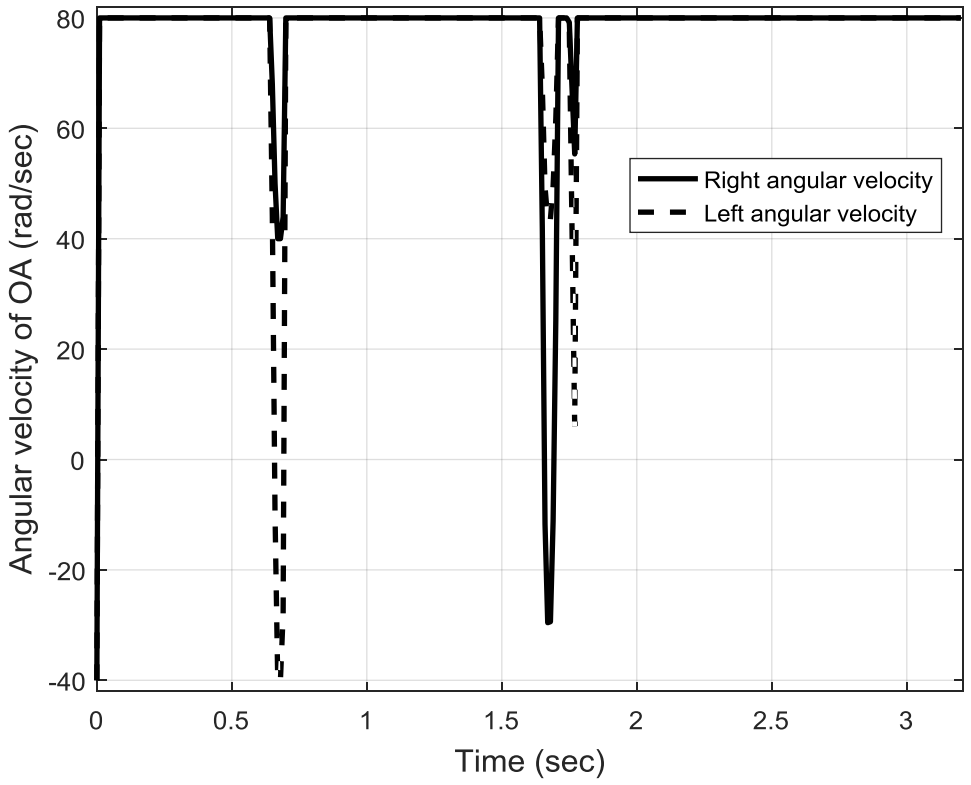



Fig. 6.16 Angular velocity for obstacle avoidance of FIS in scenario-I.

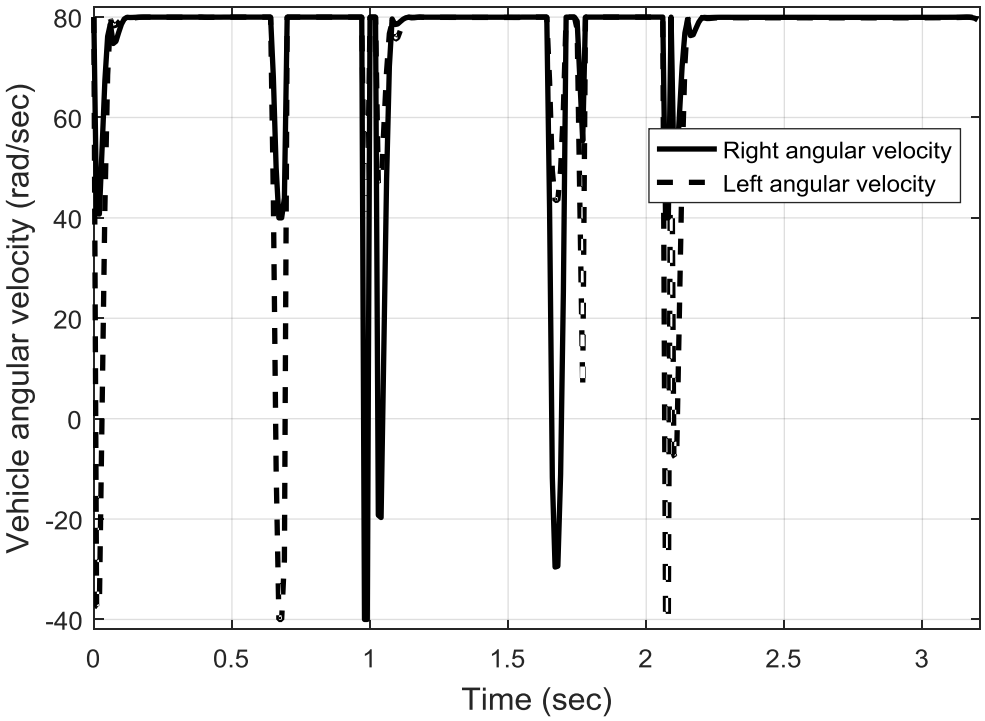



Fig. 6.17 Angular velocity of UGV in scenario-I using FIS.

The sensory information on the left, front and right distances is obtained as shown in Fig. 6.18. It is observable that the UGV can avoid obstacle no.4 by simply using only the right distance sensor. However, in accordance to obstacle no. 6, the three distances are detected. In fact, this has occurred whilst obstacle no.6 moves closely around the surrounding of the UGV. Fig. 6.19 presents the time response of the obstacle sensing indicator; this transmits a signal to the switching mechanism to select between the two FIS controllers of the target reaching and obstacle avoidance. It has a value of '1' when an obstacle is detected and the FIS of obstacle avoidance will be activated accordingly. Otherwise, the sensing indicator will have a value of '0', which means the path is clear of obstacles. Hence, the FIS of the target reaching will be operated to guide the UGV to its destination.

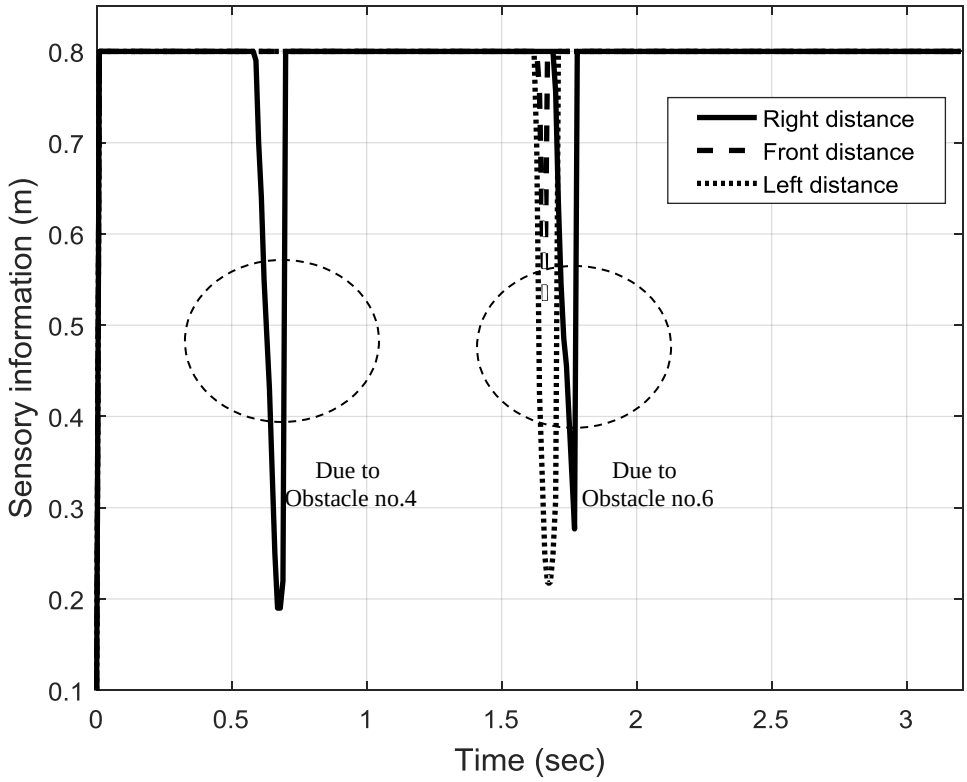



Fig. 6.18 Sensory information of three sensors in scenario-I using FIS.

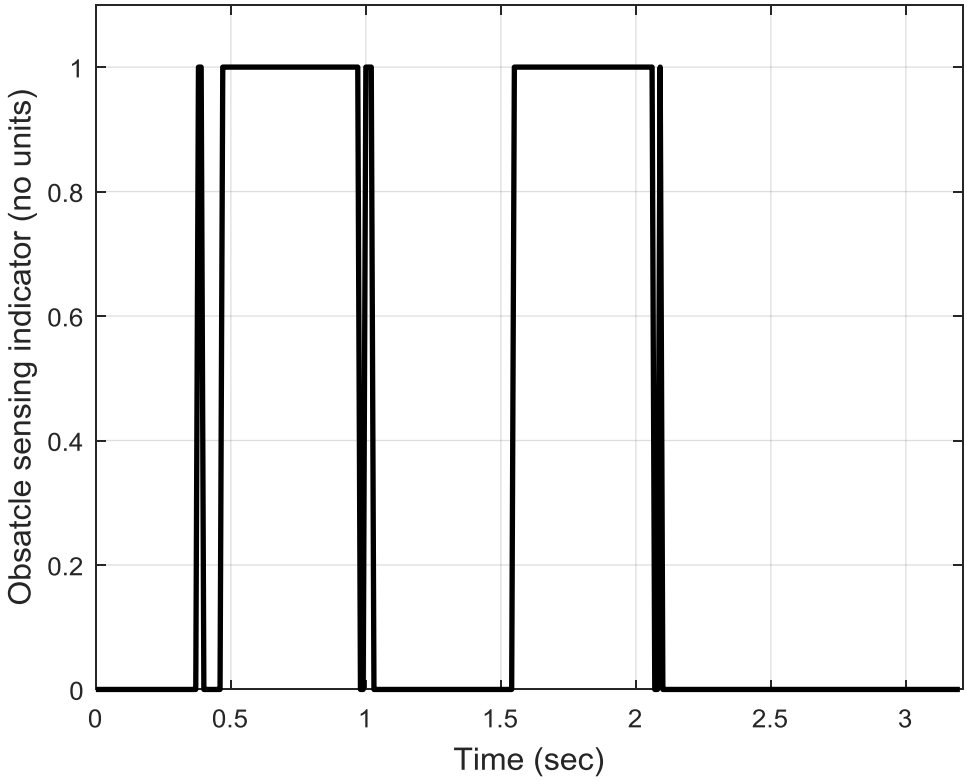



Fig. 6.19 Obstacle sensing indicator in scenario-I using FIS.

To validate the basic functionality of the proposed algorithm for guiding the UGV to track the destination, the differences between the target and actual coordinates of X and Y-axes are demonstrated in Fig. 6.20 and Fig. 6.21, respectively. The curves on both graphs illustrate that the movement of the UGV is achieved successfully toward the target from the initial position. Despite of the larger error at the commencing of the movement, it is important to notice the self-regulatory behaviour of the proposed navigation algorithm. As soon as the tracking error is quite large due to the far distance between the initial position and the target point, the algorithm automatically adjusts the UGV's orientation to a proper direction so that it heads back toward the target whilst avoiding obstacles. The tracking error of both of X and Y coordinates is shown in Fig. 6.22.

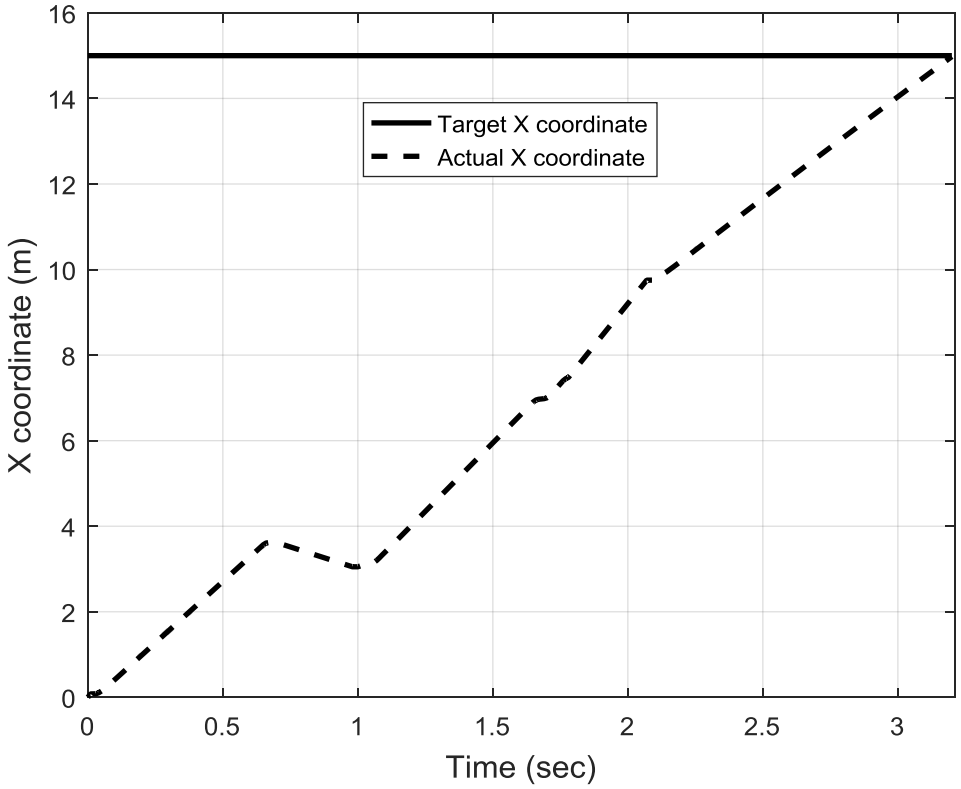



Fig. 6.20 X-coordinate of the UGV whilst navigation in scenario-I using FIS.

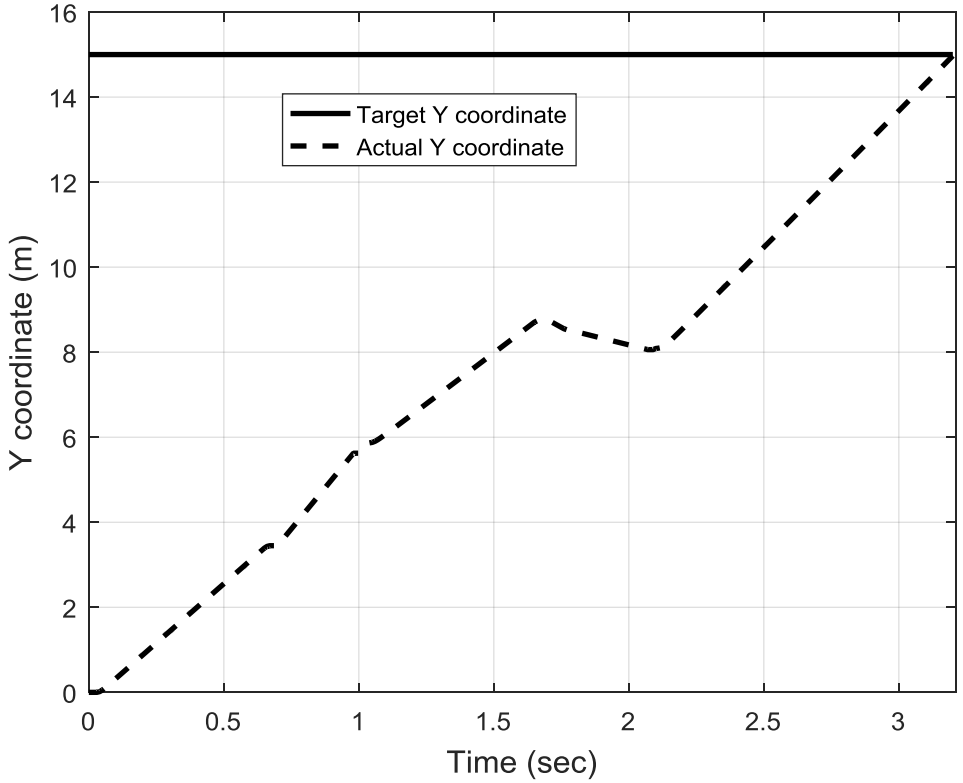



Fig. 6.21 Y-coordinate of the UGV whilst navigation in scenario-I based on FIS.

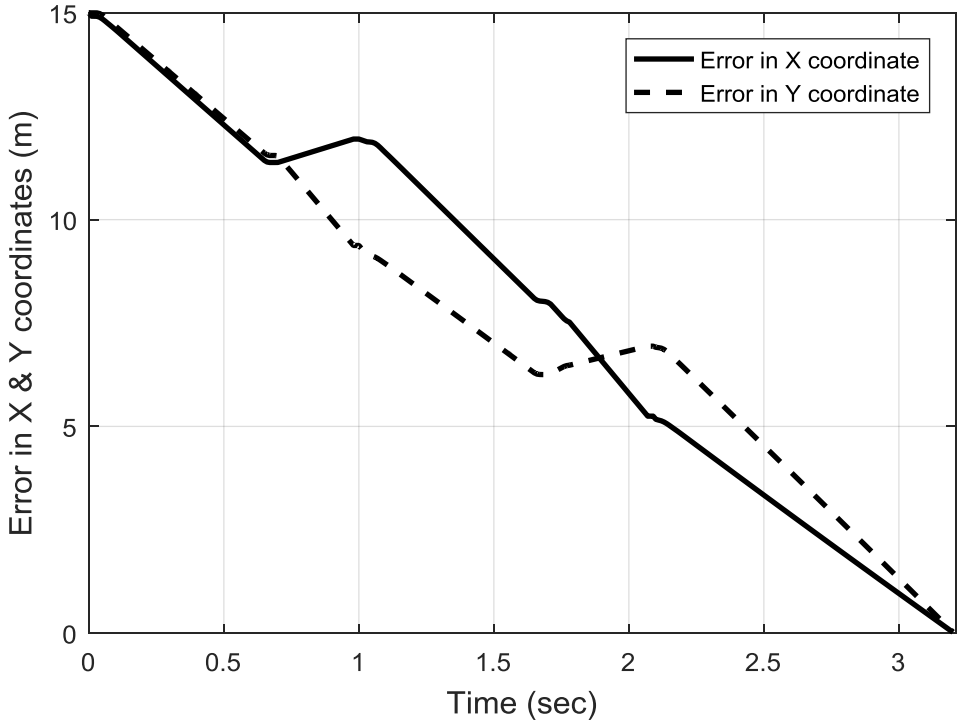



Fig. 6.22 Tracking error in X and Y coordinates of UGV in scenario-I using FIS.

### *6.6.2 Scenario-II: Dynamic obstacles with similar sizes but different velocities*

In this scenario as shown in Fig. 6.23, the UGV navigation platform is conducted in an environment where the obstacles are moving at different velocities. Obstacles '1' and '2' are moving at a speed of 2.65 m/s, obstacles '3' and '4' are moving at a speed of 1.8 m/s, and obstacles '5' and '6' are moving at a speed of 4.6 m/s. All the six moving obstacles are successfully avoided by the UGV whilst it has travelled from the starting point towards the destination coordinates. Interestingly, although this scenario is expected to be more challenge than the antecedent scenario based on the hypothesis of random velocities, it can be observed that the generated path is shorter regardless of the scenario complexity. Fig. 6.24 demonstrates the changing in the orientation of the UGV whilst avoiding obstacles. The linear velocity profile of UGV is shown in Fig. 6.25. The target reaching and obstacle avoidance FIS controllers have provided the driving wheels with the necessary angular velocities to change the direction of the UGV and avoid collisions with the moving obstacles as shown in Fig. 6.26 and Fig. 6.27, respectively. The outputs of both FIS controllers are associated through the switching mechanism. The outputs of the switching mechanism are applied to the UGV's wheels to find the turning angle of the UGV as shown in Fig. 6.28. The decisions are made according to the distances between the left, the right and the front of the UGV with respect to the approaching obstacles. In this scenario, the elapsed time of the UGV journey from the starting point to the destination coordinates is equal to 2.95 seconds.

The random movement of obstacles makes the obstacles widespread at different positions in the environment. This is rather a surprising finding to have only one obstacle that could be attributed to the UGV navigation. Therefore, it seems that this scenario may have the shortest path. Fig. 6.29 demonstrates the sensory information of the three ultrasonic sensors. In additional, the obstacle sensing indicator is depicted in Fig. 6.30. It introduces a signal to switching mechanism within the obstacle detection range. The X and Y coordinates of the UGV whilst moving in the 2- dimensional grid are presented in Fig. 6.31 and Fig. 6.32, respectively. Accordingly, the corresponding tracking error is introduced in Fig. 6.33 that ascertains that the UGV has reached its destination.

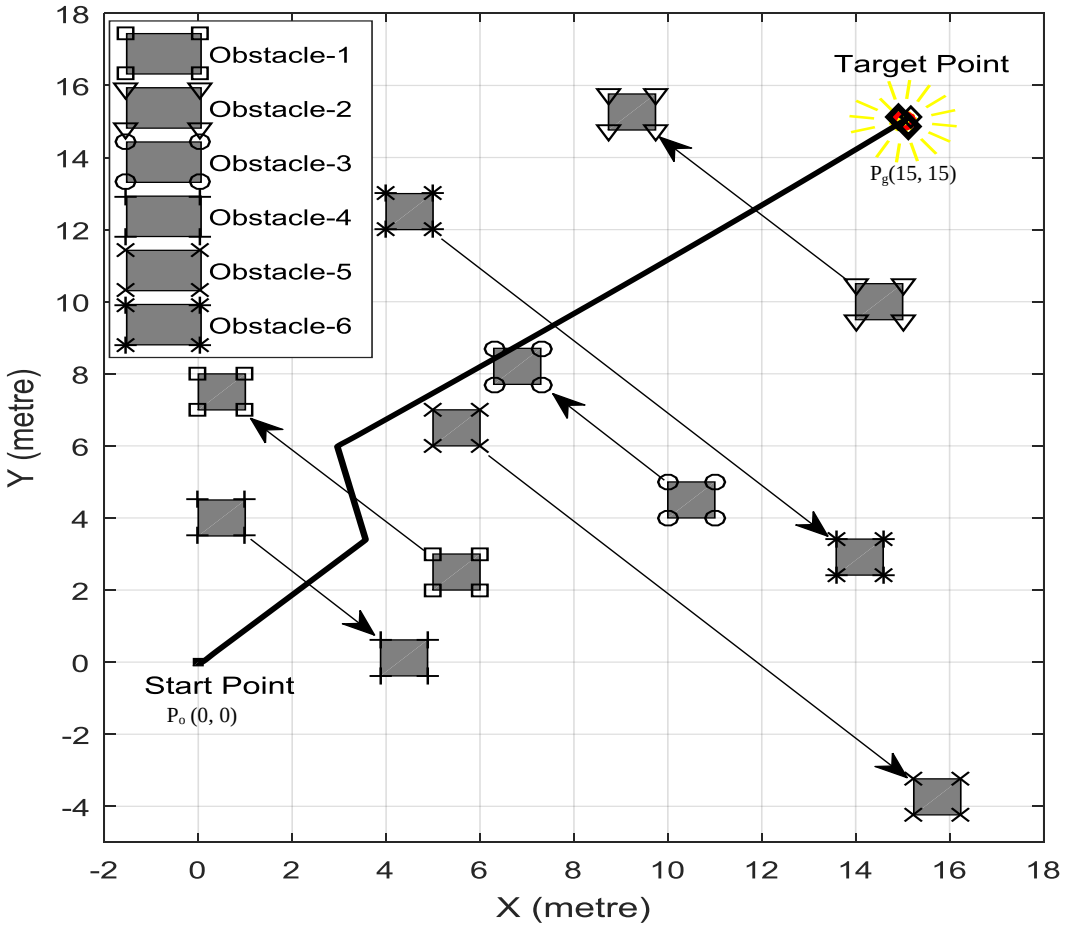



Fig. 6.23 Navigation platform and topology for scenario-II using FIS.

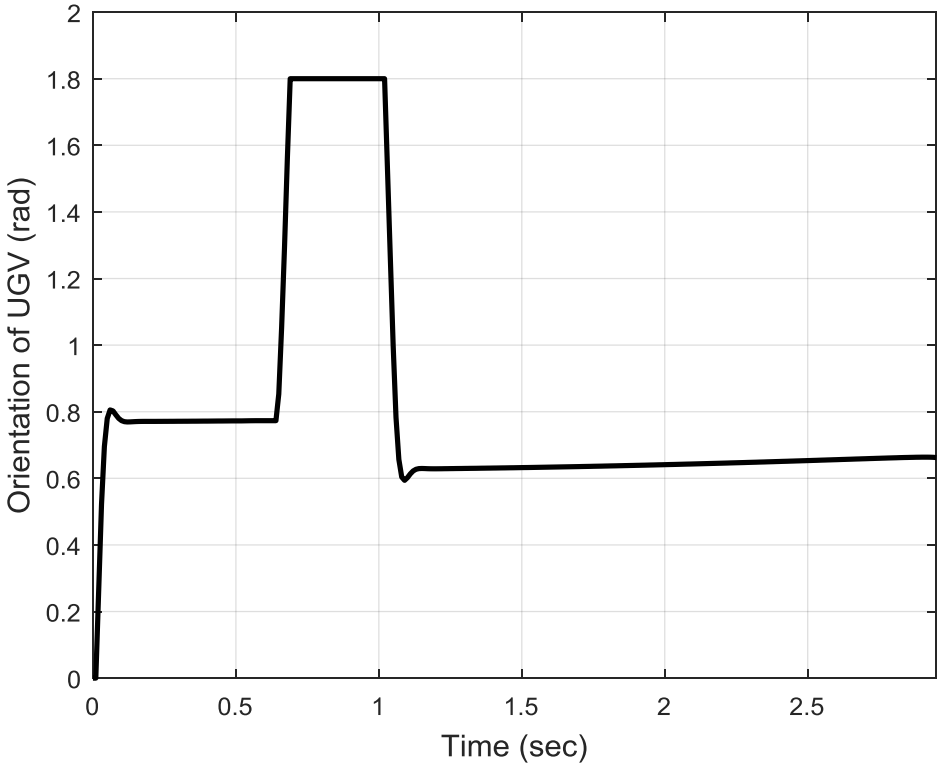



Fig. 6.24 Orientation of UGV whilst navigation in scenario-II using FIS.

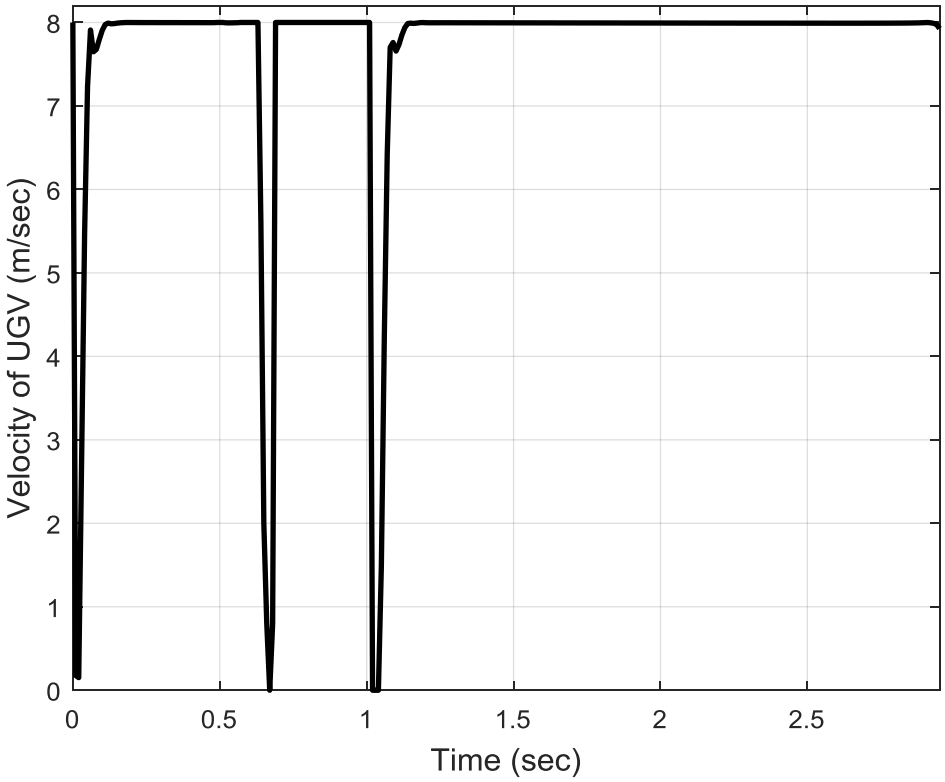



Fig. 6.25 Linear velocity of UGV whilst navigation in scenario-II using FIS.

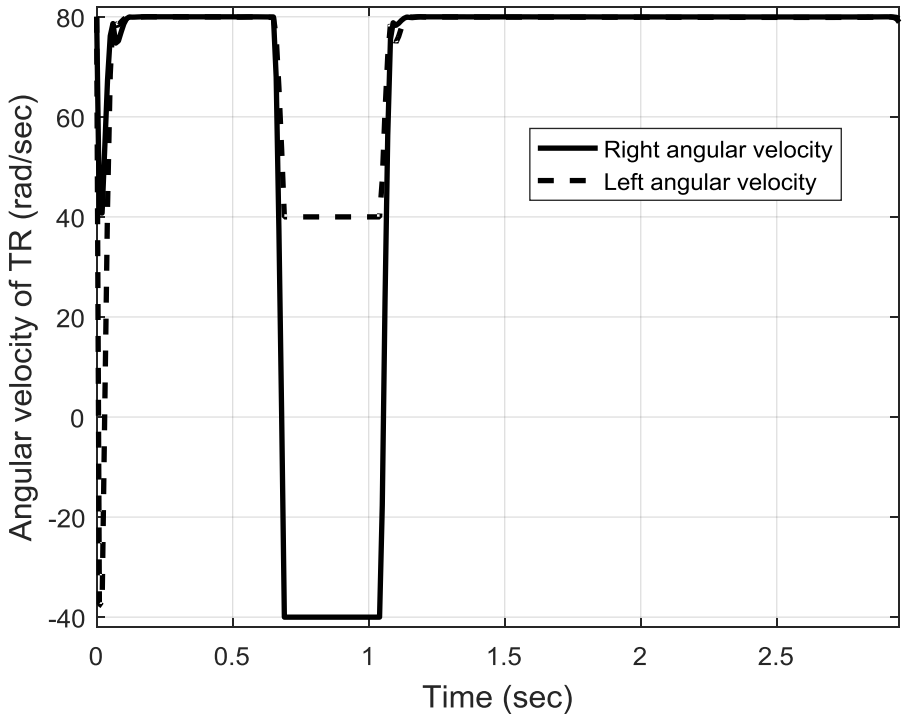



Fig. 6.26 Angular velocity for target reaching of FIS in scenario-II.

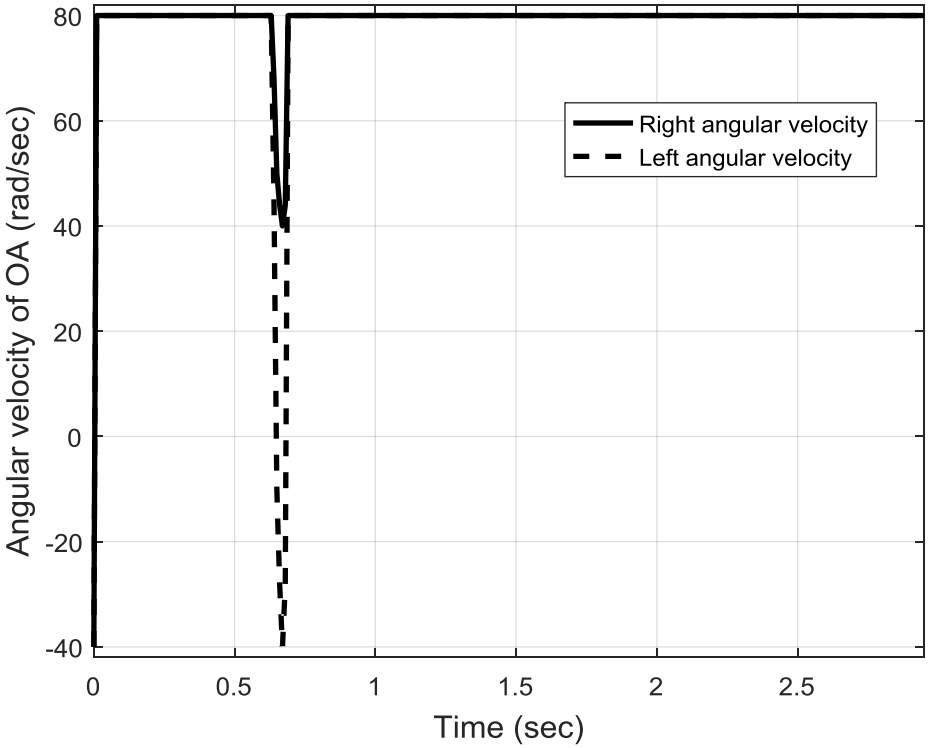



Fig. 6.27 Angular velocity for obstacle avoidance of FIS in scenario-II.

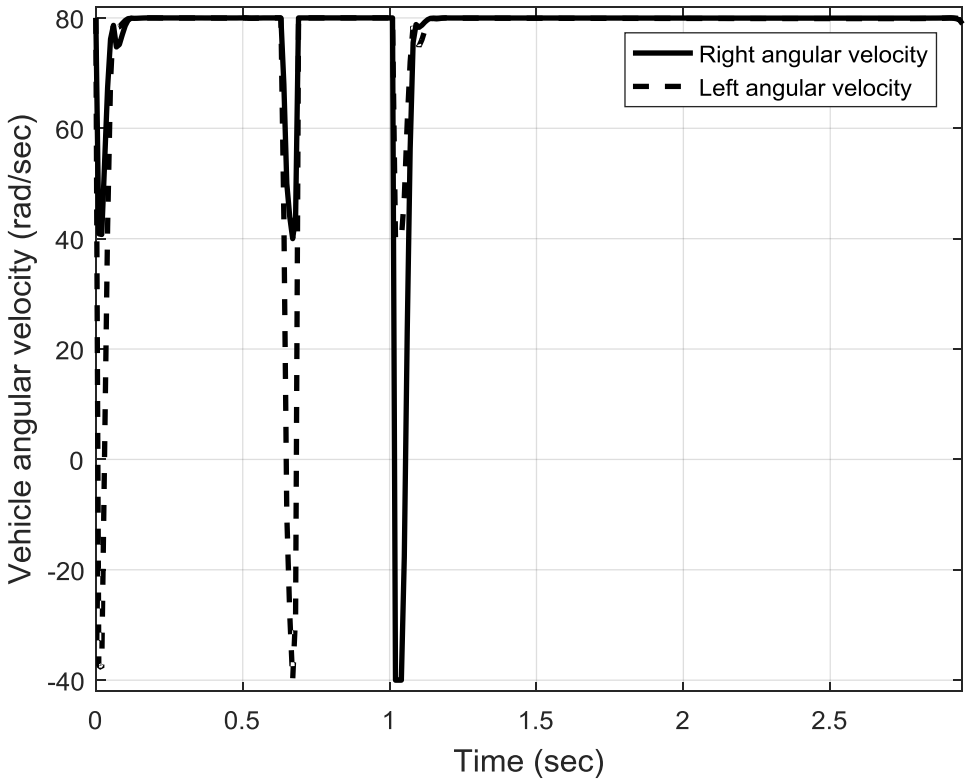



Fig. 6.28 Angular velocity of UGV whilst navigation in scenario-II using FIS.

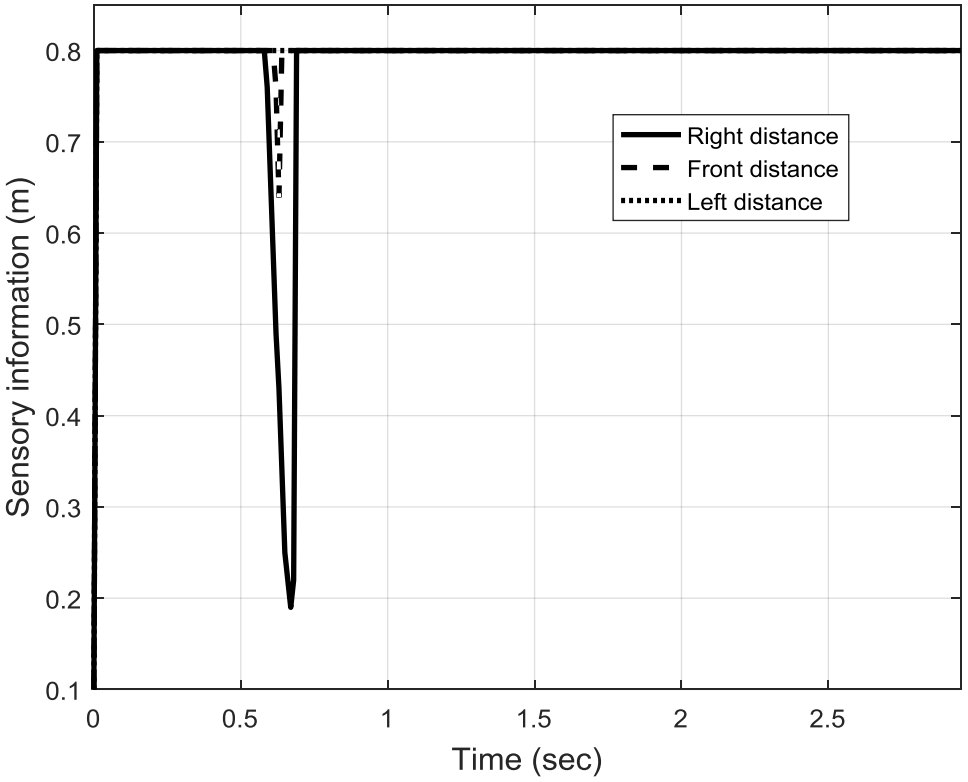



Fig. 6.29 Sensory information of three sensors in scenario-II using FIS.

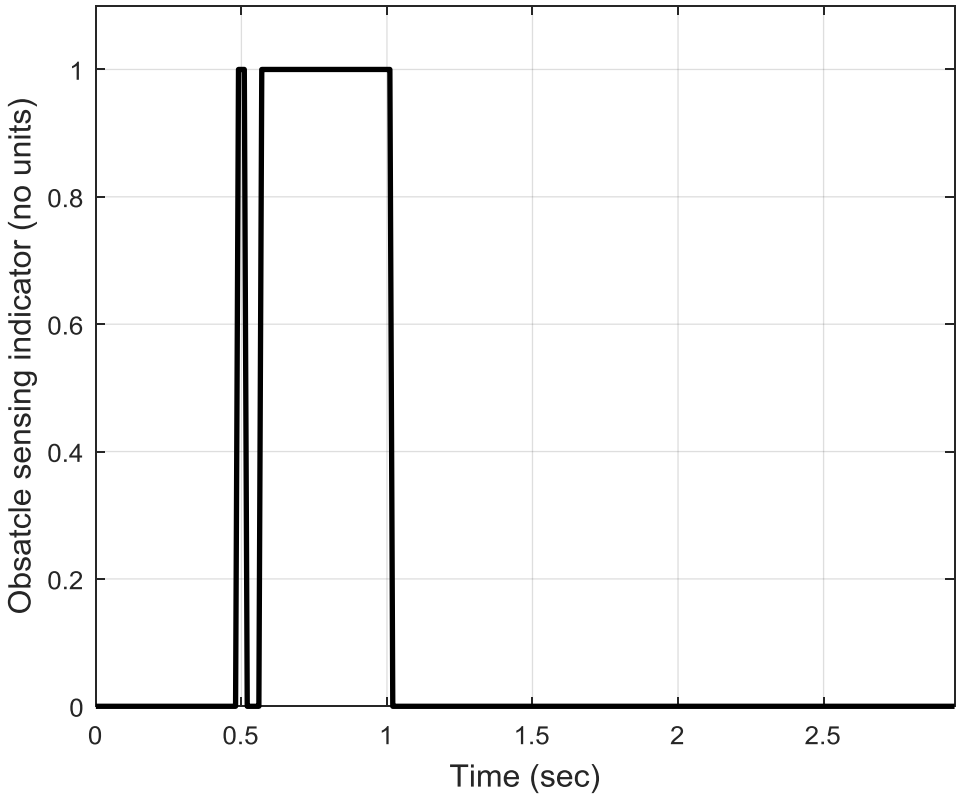



Fig. 6.30 Obstacle sensing indicator whilst navigation in scenario-II using FIS.

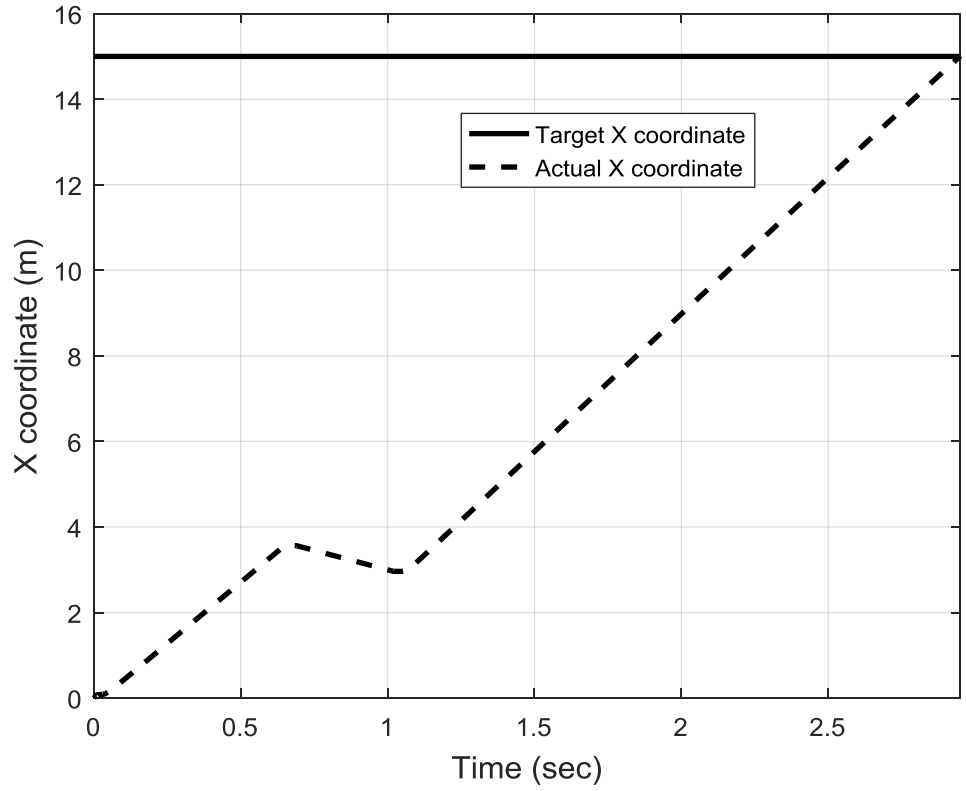



Fig. 6.31 X-coordinate of UGV whilst navigation in scenario-II using FIS.

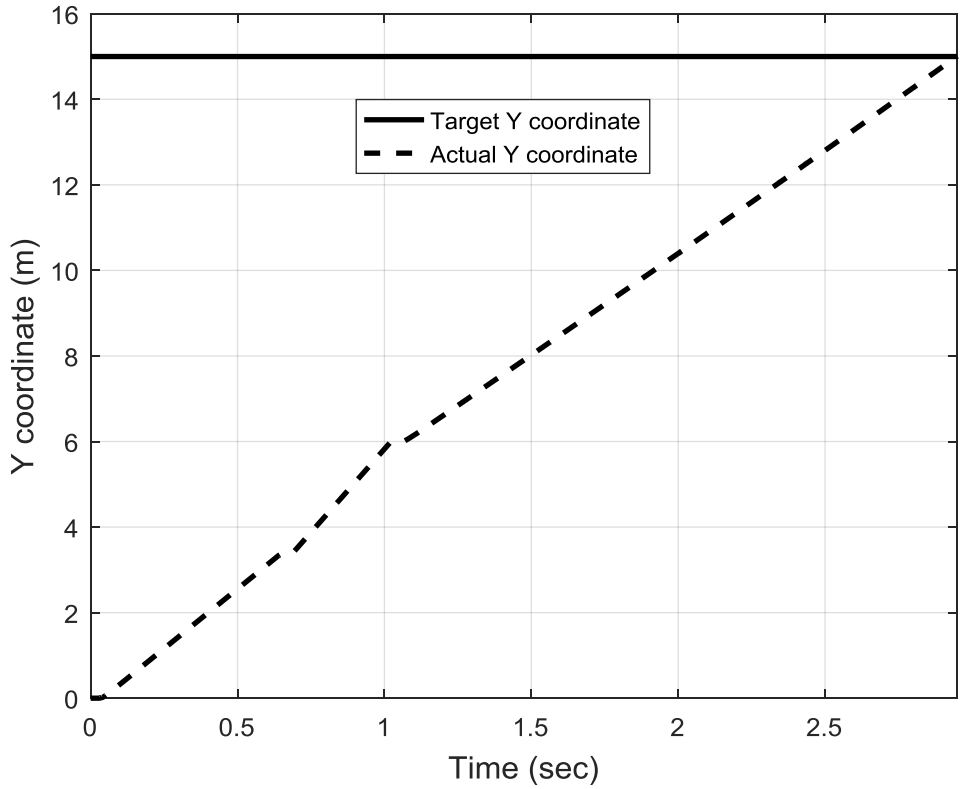



Fig. 6.32 Y-coordinate of UGV whilst navigation in scenario-II using FIS.

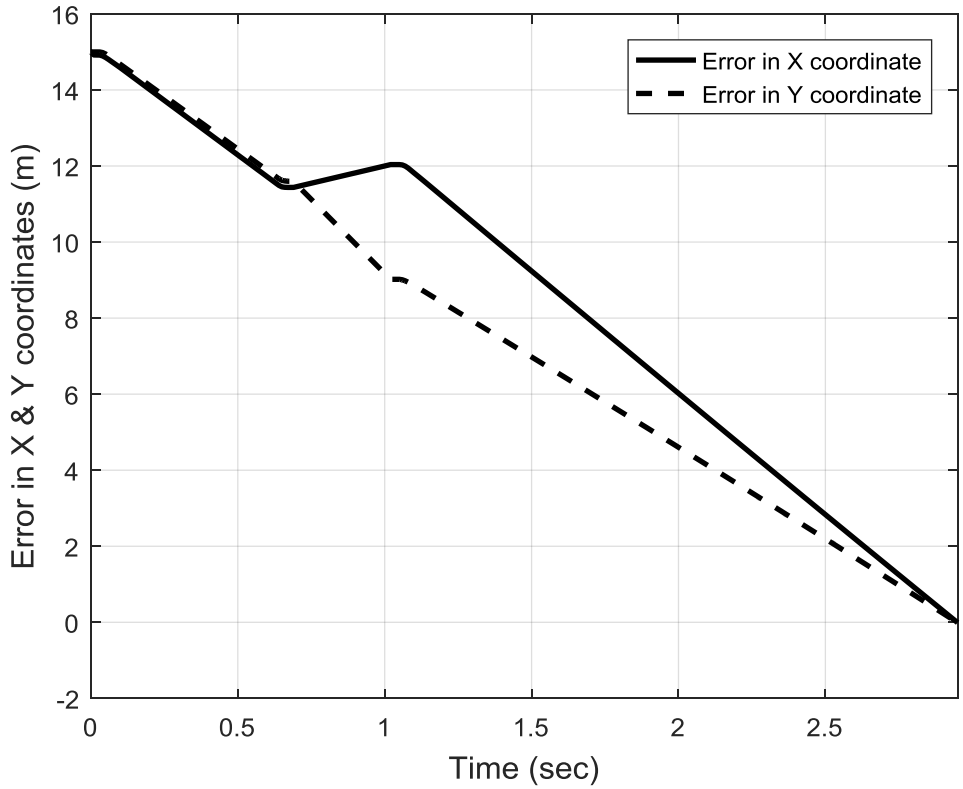



Fig. 6.33 Tracking error in X and Y coordinates of UGV in scenario-II using FIS.

### 6.6.3 Scenario-III: Dynamic obstacles with different velocities and sizes

In this scenario, to validate the proposed algorithm against a more complex environment, the UGV navigation platform is operated with six moving obstacles that are varied in their locomotion. The velocity of the obstacles no.3 and no.5 equals to 1.7 m/s. All the other obstacles move at velocity of 2.55 m/s. Fig. 6.34 demonstrates that the moving obstacles are avoided and the UGV reached the target. The OA-FIS controller has made the UGV avoided the moving obstacles by making decisions on where to change the direction of the UGV. That allowed the UGV to prevent collisions with the obstacles. It is observable that obstacle '1' left its original position towards a new position in the workspace. Therefore, its original place has become free. The TR-FIS controller is activated when there is no obstacle approaching the UGV. The target destination in this scenario is reached successfully by the UGV after avoiding all the moving obstacles and the elapsed time equals to 3.58 seconds. Fig. 6.35 illustrates the changing in the direction of the UGV whilst avoiding obstacles. The linear velocity profile of the UGV is shown in Fig. 6.36. It is noticed that the size of the obstacles has no influence over the performance of the controllers even though larger objects limited the free space in the environment. However, this is accomplished by the obtained distance readings from the UGV sensors. In practical, when the UGV is near a movable obstacle, it must be ready to react responsively to take care of the sudden motion of that obstacle. Therefore, the velocity response is decreased when the UGV enters the deceleration zone.

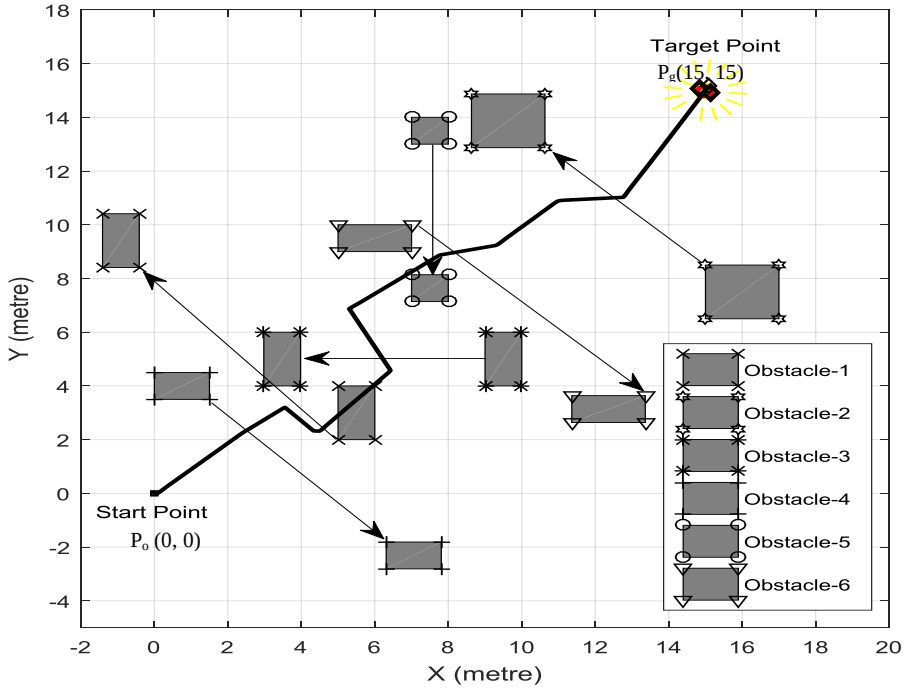



Fig. 6.34 Navigation platform and topology for scenario-III using FIS.

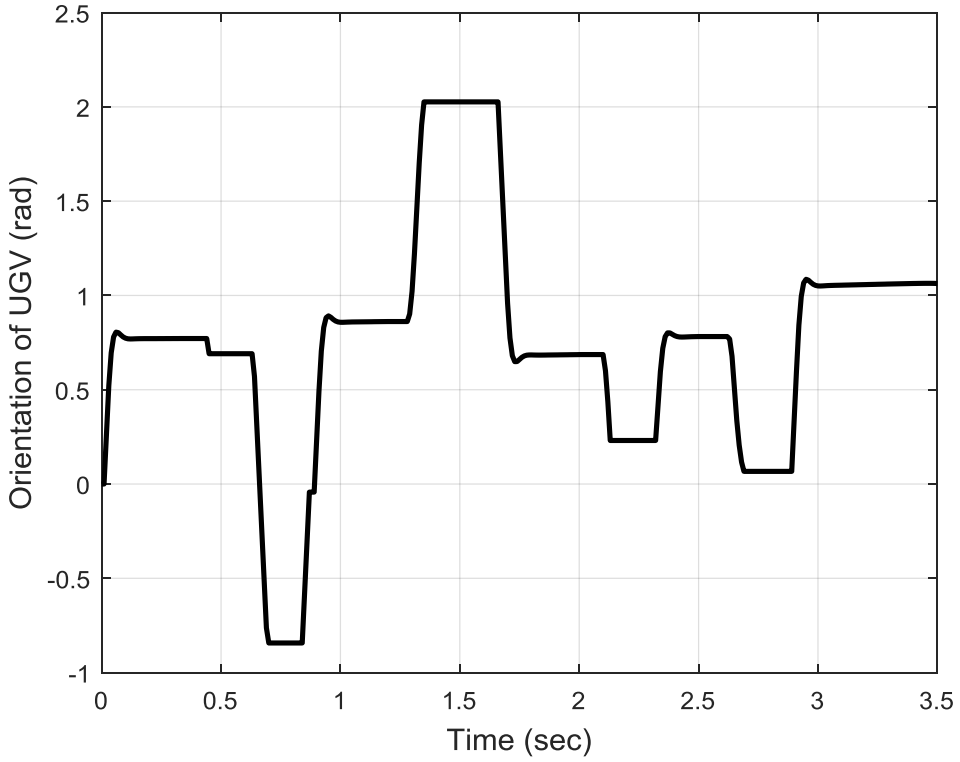



Fig. 6.35 Orientation of UGV whilst navigation in scenario-III using FIS.

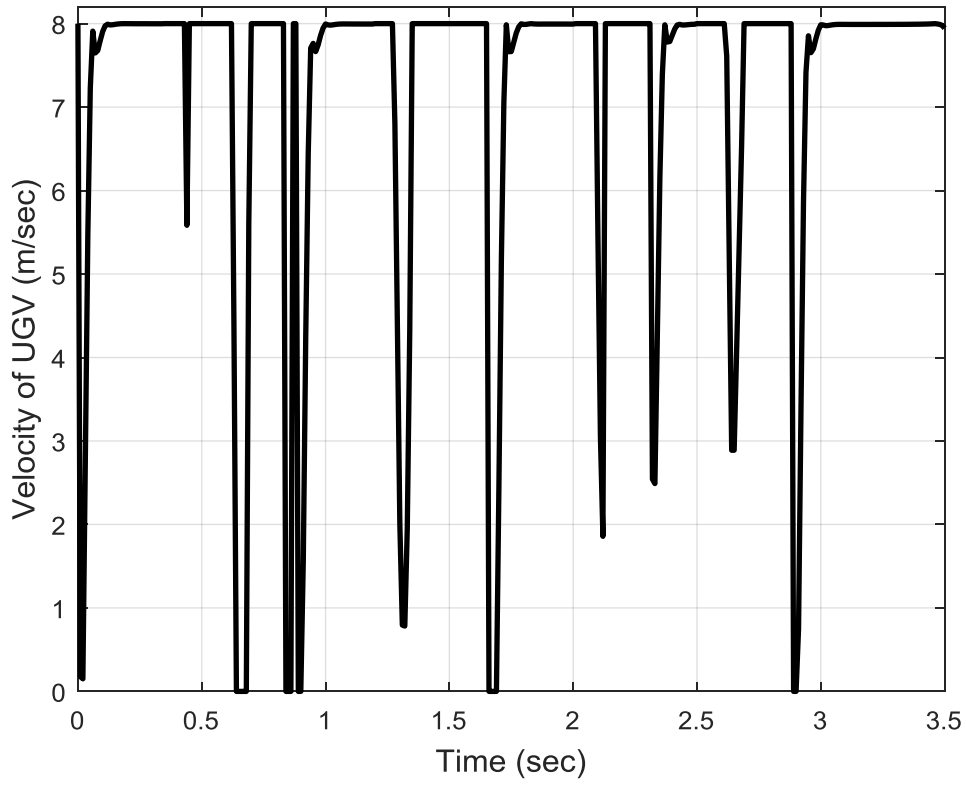



Fig. 6.36 Linear velocity of UGV whilst navigation in scenario-III using FIS.

Similarly, the angular velocities of the target reaching and obstacle avoidance FIS controllers are shown in Figs. 6.37 and 6.38. The total angular velocity that derives the UGV is presented in Fig. 6.39. In comparison to the previous scenarios, it can be observed that the UGV consistently confronts obstacles from the initial to the target point. Regardless, the proposed algorithm demonstrates a superiority of achieving the navigation successfully without colliding with obstacles. Fig. 6.40 shows the sensory information for this scenario. Consequently, the obstacle-sensing indicator is depicted in Fig. 6.41. Although the six obstacles have obstructed the path planning of the UGV, the UGV has demonstrated a quite decent efficacy in completing its task. Moreover, the coordinates of X and Y-axes are given in Figs. 6.42 and 6.43. Accordingly, the tracking error for both coordinates is introduced in Fig. 6.44 to demonstrate that the UGV has reached the destination. Hence, the validity of the proposed algorithm has been proved.

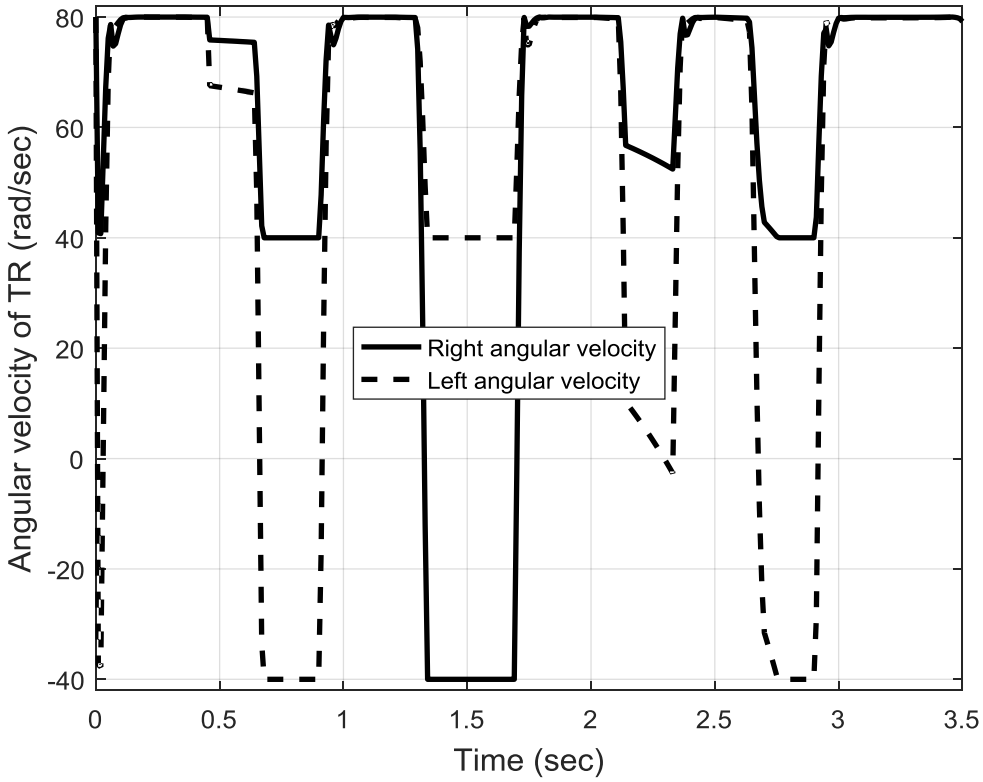



Fig. 6.37 Angular velocity for target reaching of FIS in scenario-III.

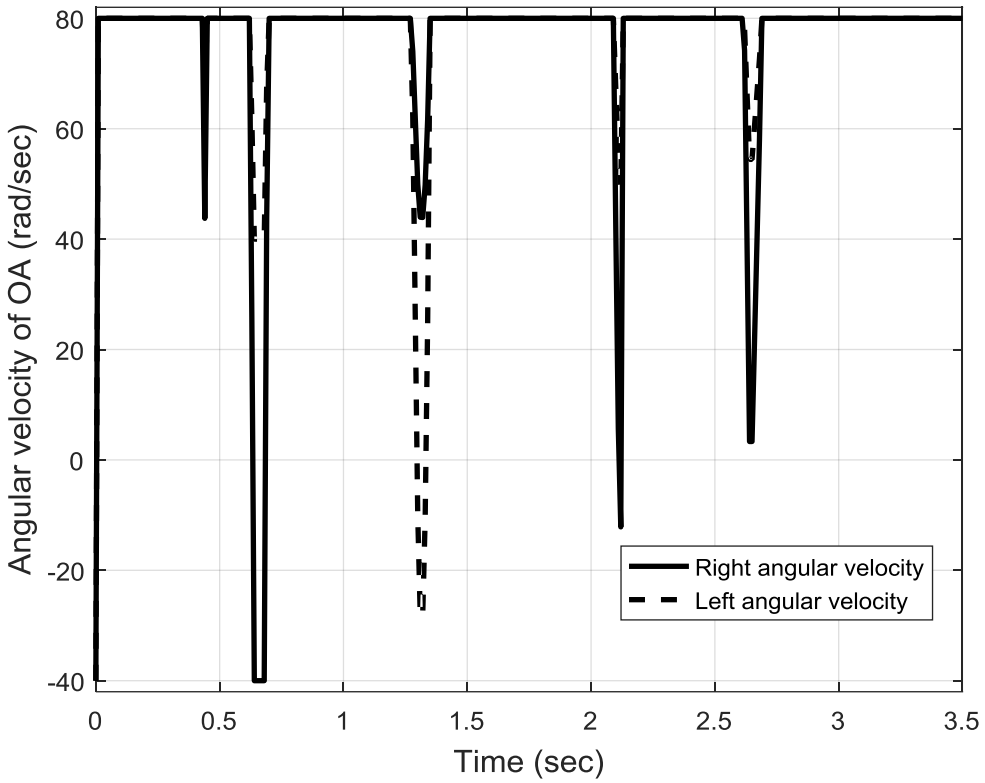



Fig. 6.38 Angular velocity for obstacle avoidance of FIS in scenario-III.

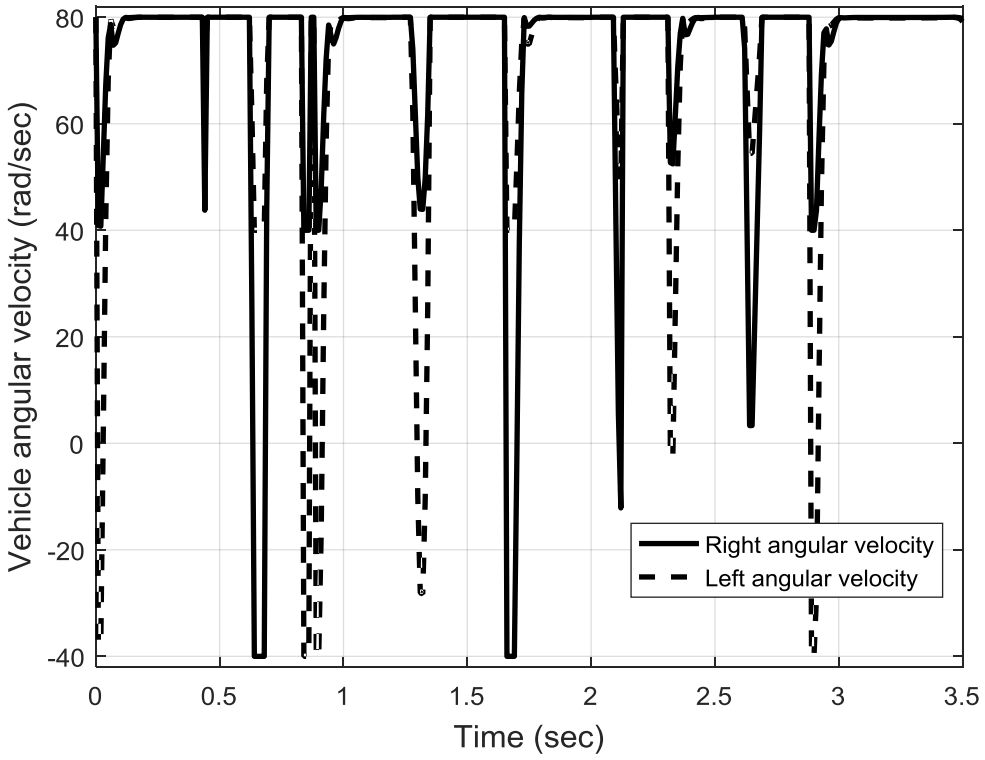



Fig. 6.39 Angular velocity of UGV whilst navigation in scenario-III using FIS.

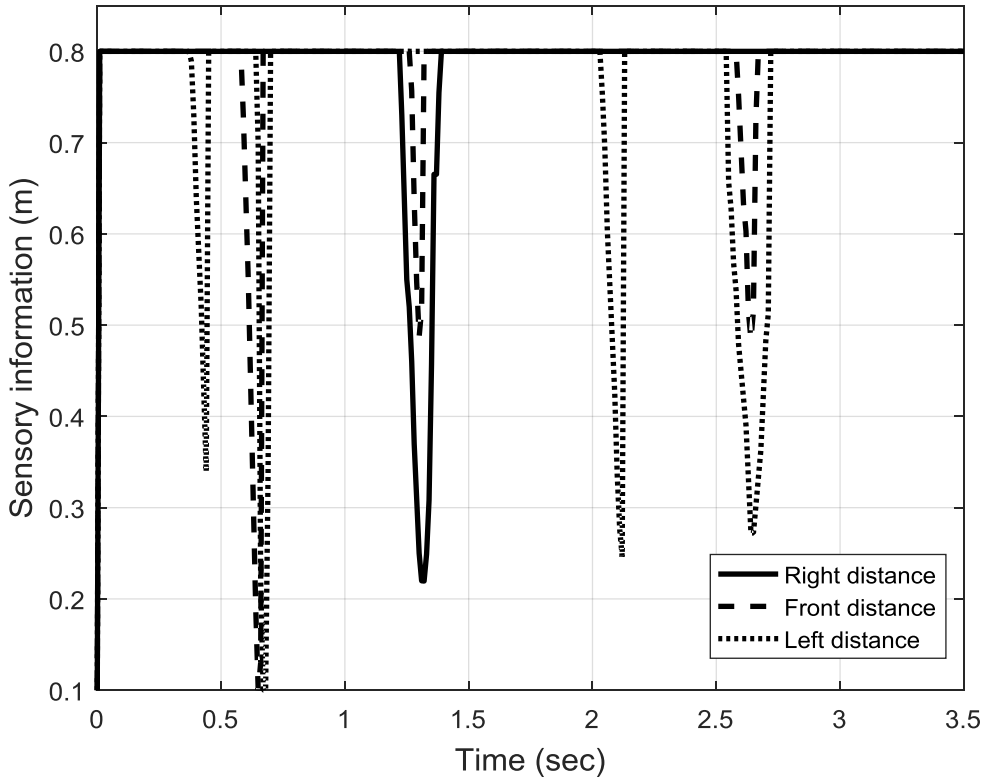



Fig. 6.40 Sensory information of three sensors of the UGV in scenario-III using FIS.

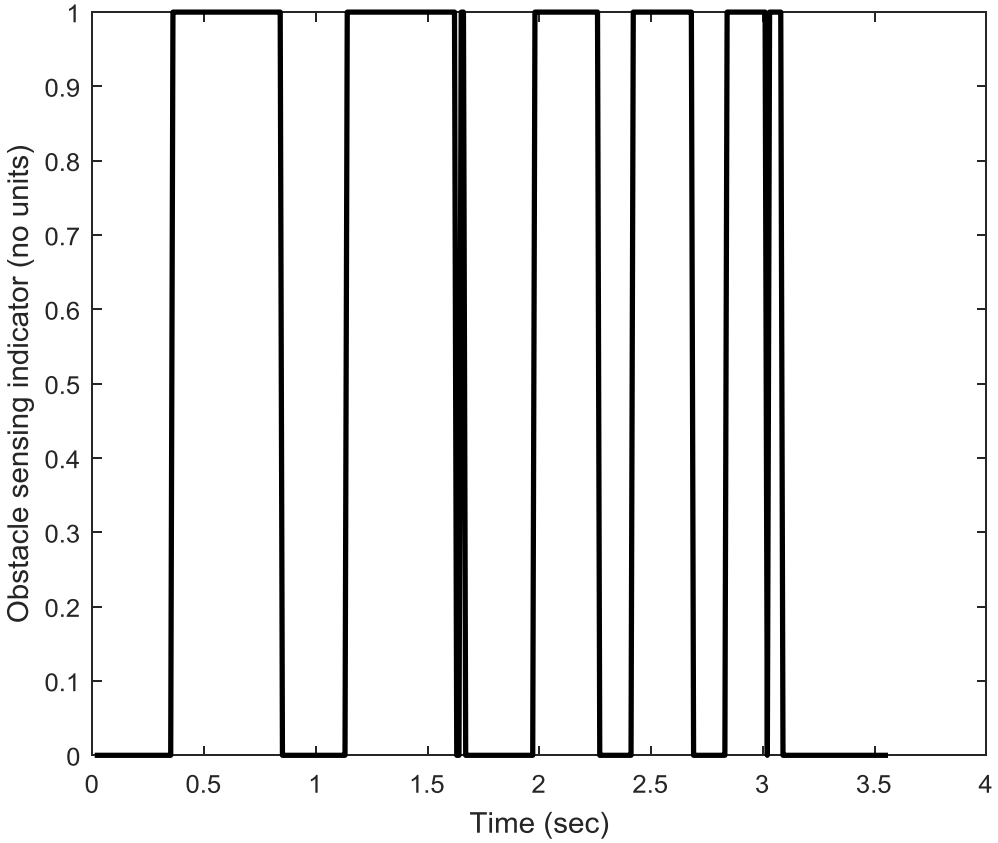



Fig. 6.41 Obstacle sensing indicator whilst navigation in scenario-III using FIS.

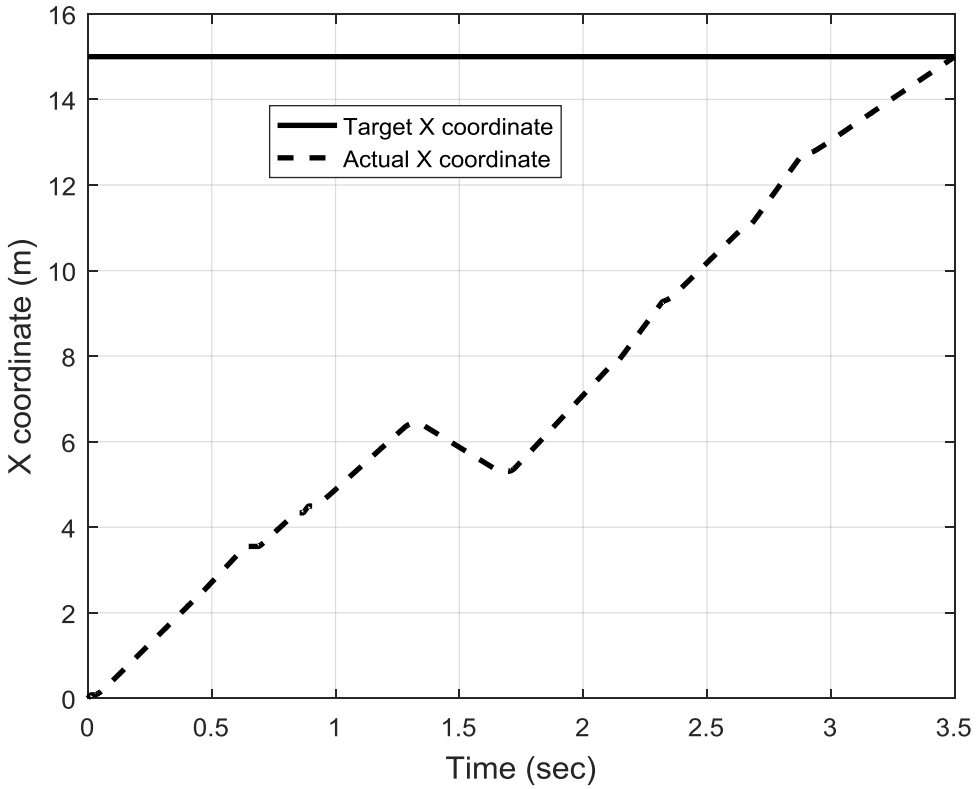



Fig. 6.42 X-coordinate of UGV whilst navigation in scenario-III using FIS.

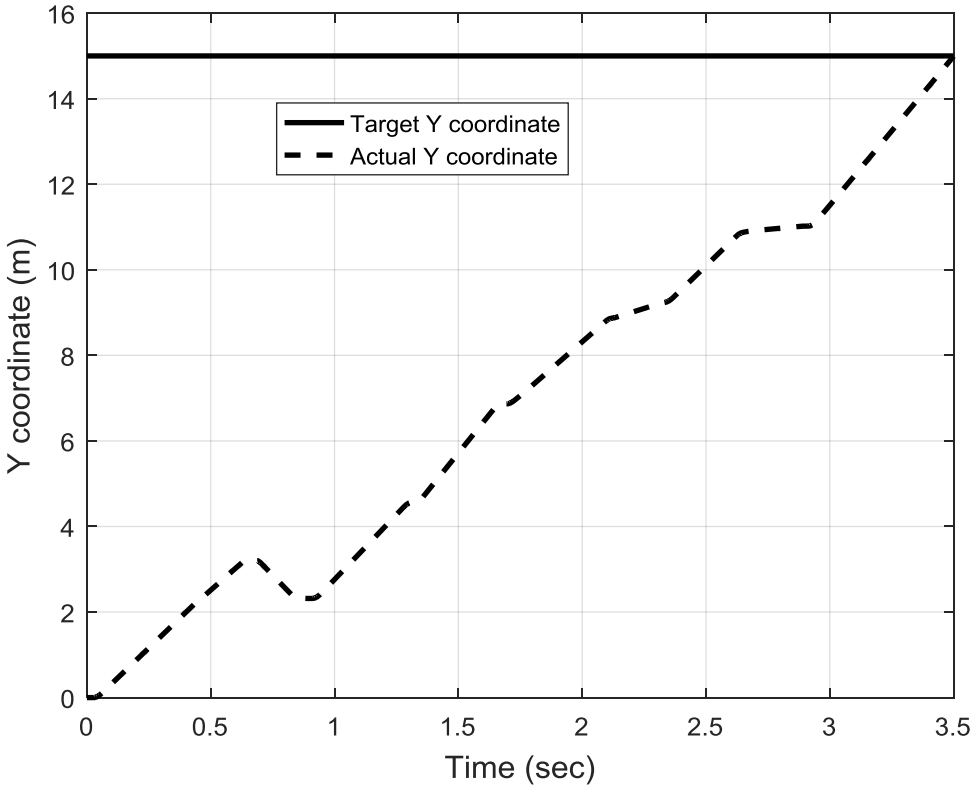



Fig. 6.43 Y-coordinate of UGV whilst navigation in scenario-III using FIS.

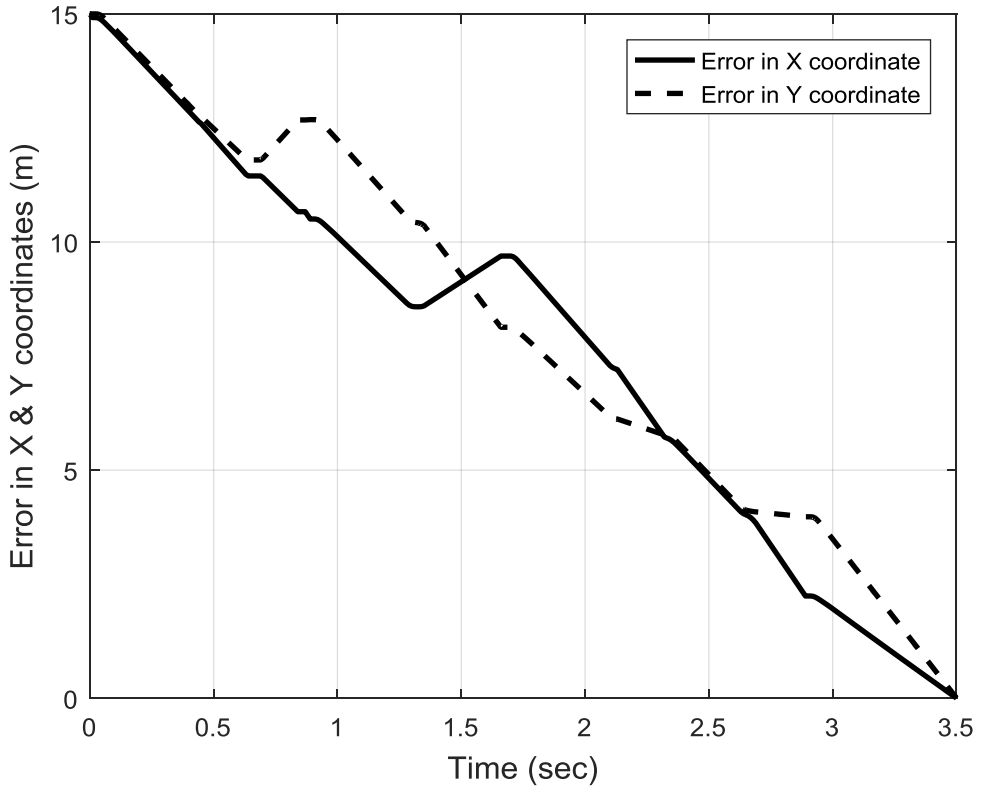



Fig. 6.44 Tracking error in X and Y coordinates of UGV in scenario-III using FIS.

### *6.6.4 Scenario-IV: Static and dynamic obstacles with different velocities and sizes*

The environment in this scenario is likewise the preceding scenario. However, the obstacle no.4 is stationary at its original position whilst the other obstacles are moving at constant velocities. Such modification makes the environment comprising of static and dynamic obstacles. This scenario has been studied to ensure that the performance is independent of the behaviour of obstacles and to guarantee the adaptation of the proposed algorithm whatsoever is the structure of the environment. Fig. 6.45 demonstrates the navigation platform of the UGV and the six obstacles. Fig. 6.46 presents the orientation of the UGV whilst navigation. It is observable that no response has occurred due to obstacle no.4 and the motion is straight at the first instance towards the destination. Fig. 6.47 introduces the linear velocity of the UGV of the switching mechanism because of obstacle avoidance and target reaching.

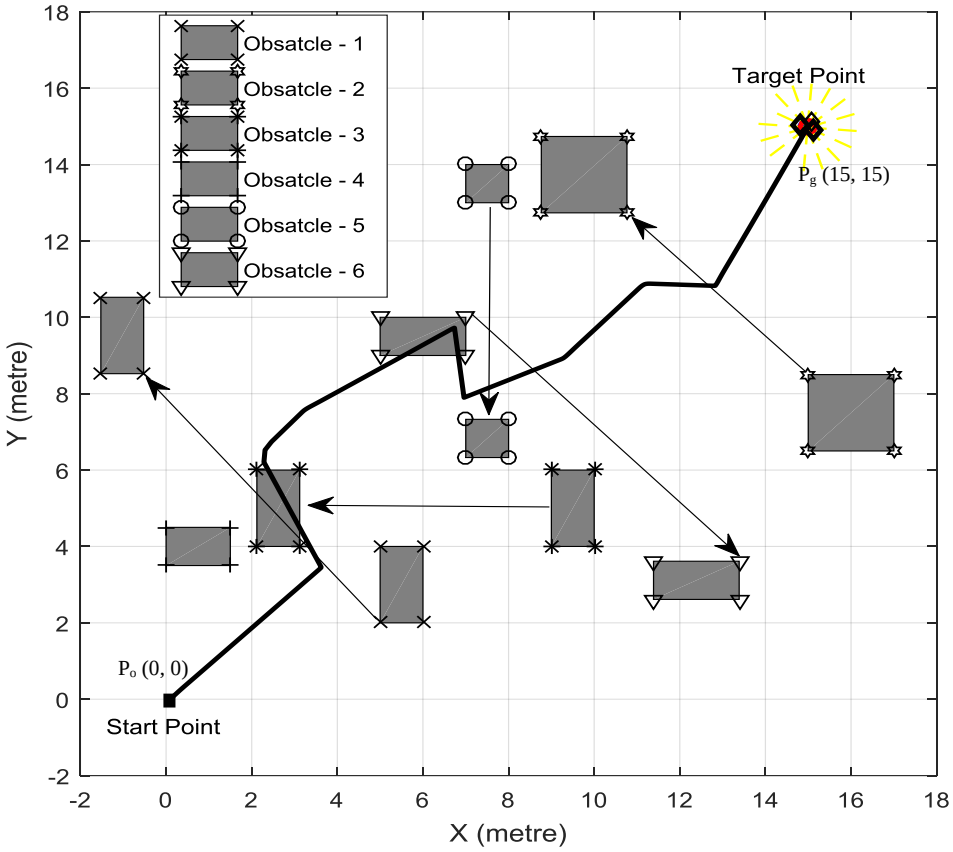



Fig. 6.45 Navigation platform and topology for scenario-IV using FIS.

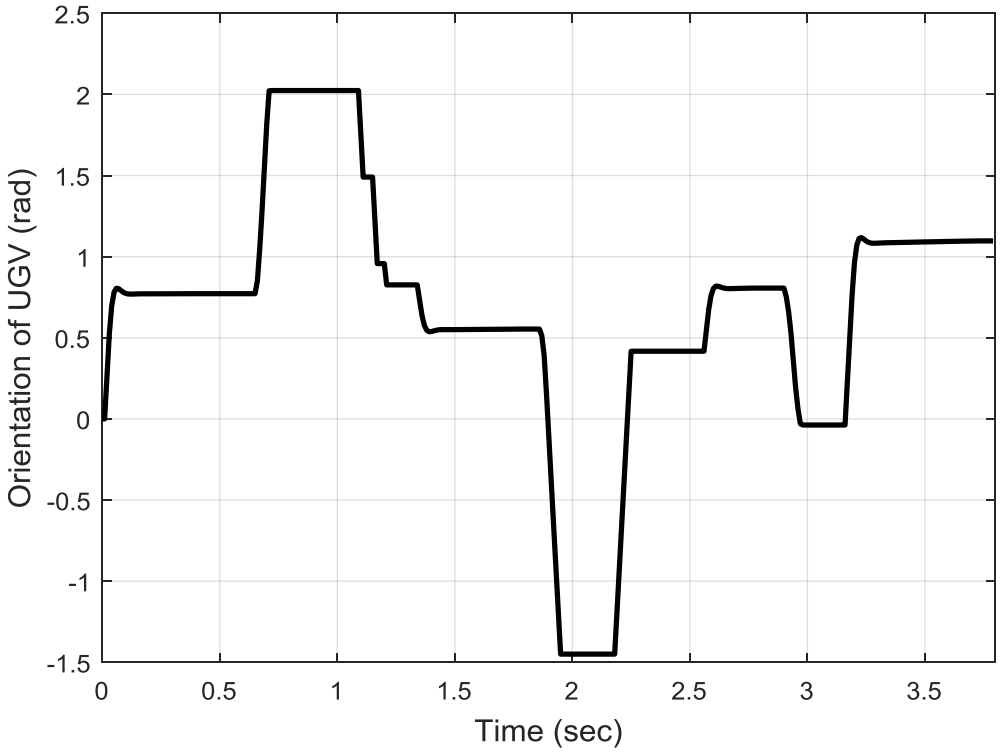



Fig. 6.46 Orientation of UGV whilst navigation in scenario-IV using FIS.

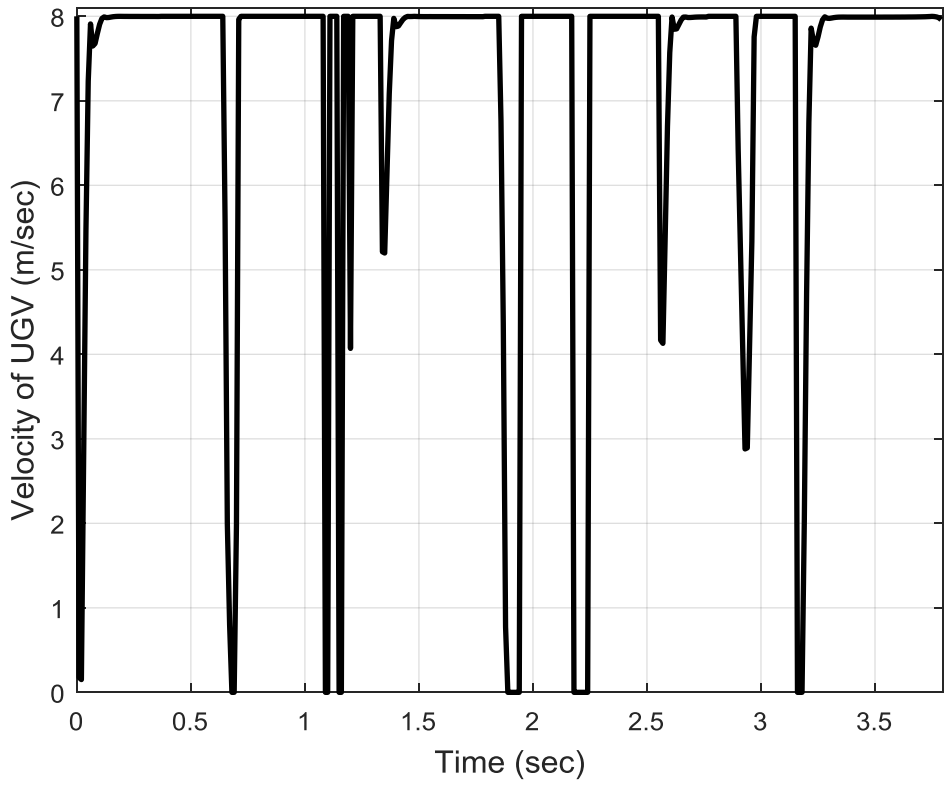



Fig. 6.47 Linear velocity of UGV whilst navigation in scenario-IV using FIS.

Figs. 6.48 and 6.49 demonstrate the angular velocity response of FIS controllers for both of target reaching and obstacle avoidance, respectively. As aforementioned, both controllers respond in accordance to the state of the working path. The final angular velocity that drives the motion of the UGV is given in Fig. 6.50. The sensory information and the corresponding obstacle sensing indicator are illustrated in Figs. 6.51 and 6.52, respectively. In addition, the navigation coordinates of X and Y axes are shown in Figs. 6.53 and 6.54. Accordingly, the obtained tracking errors of these coordinates are visualised in Fig. 6.55.

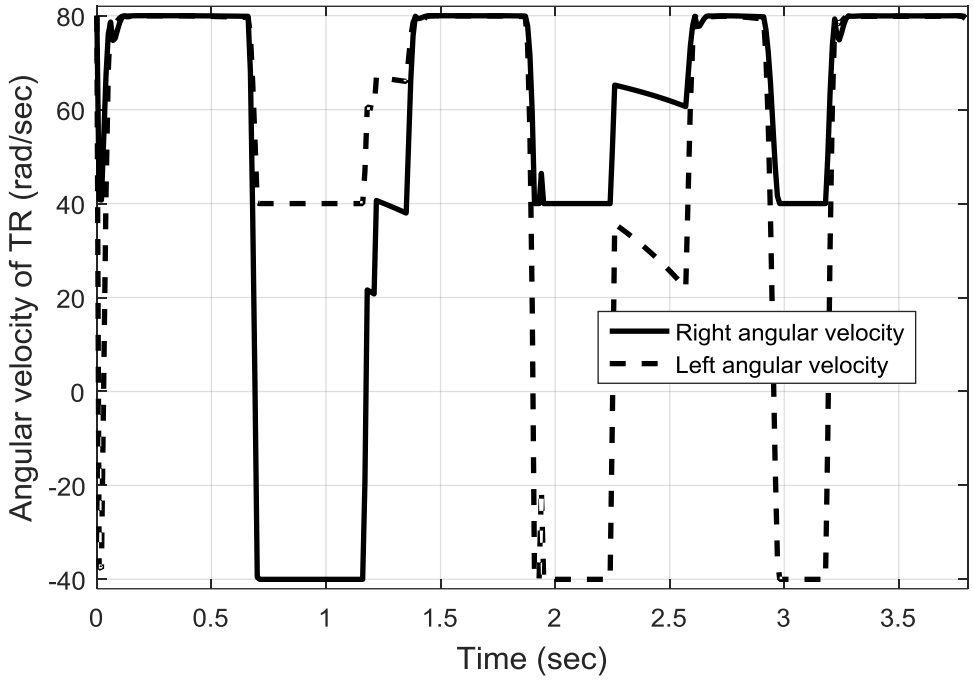



Fig. 6.48 Angular velocity for target reaching of the FIS in scenario-IV.

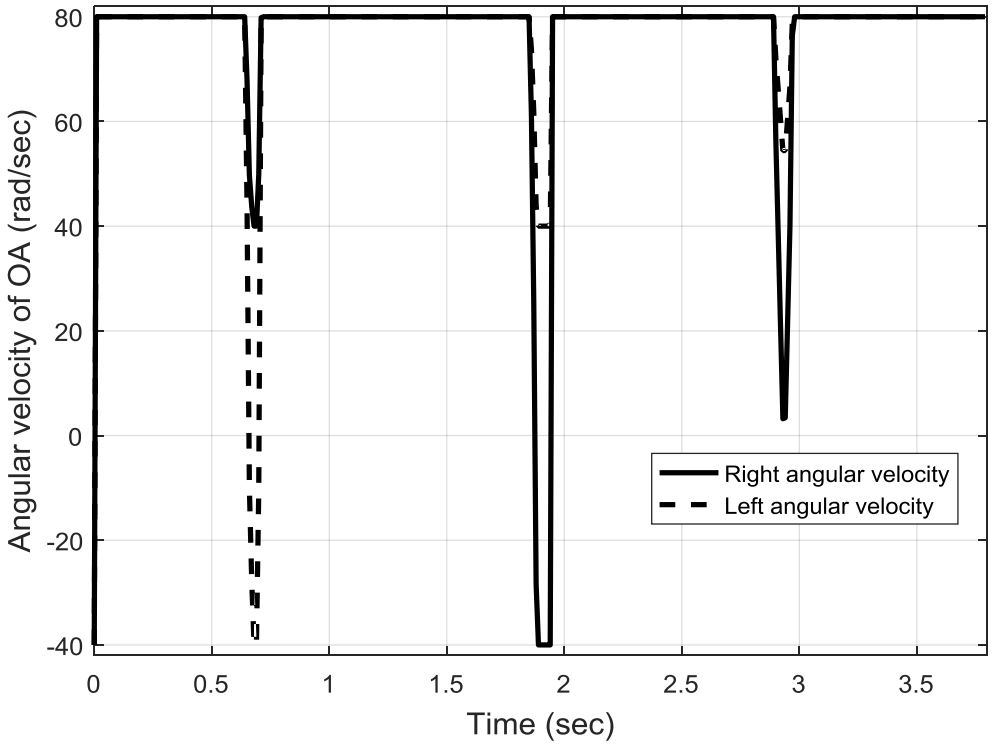



Fig. 6.49 Angular velocity for obstacle avoidance of FIS in scenario-IV.

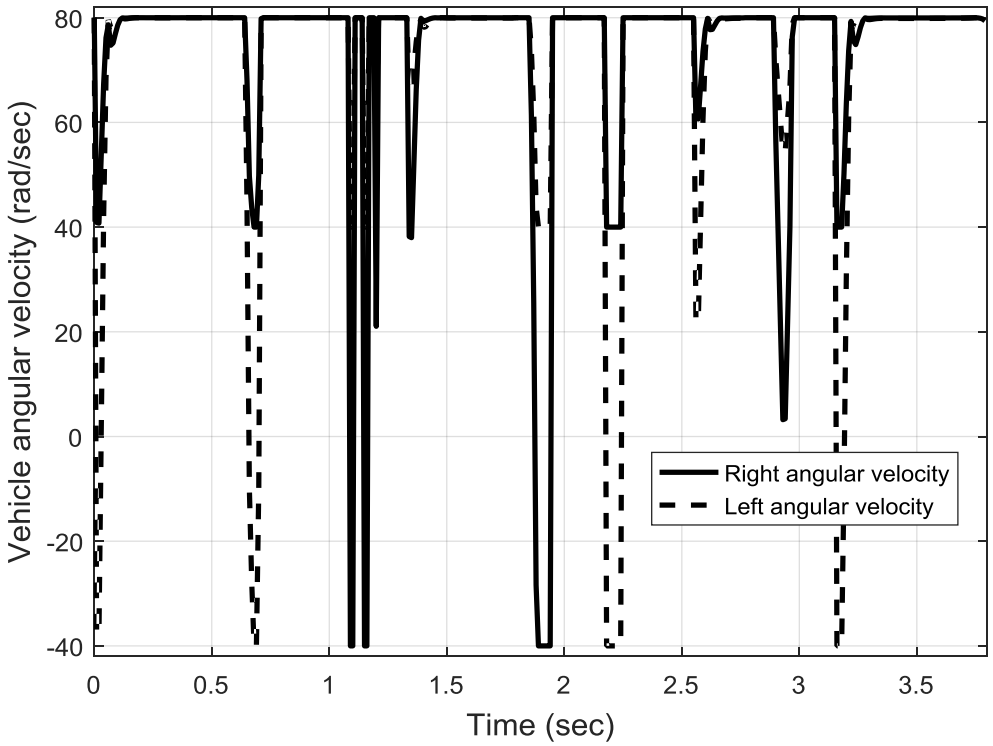



Fig. 6.50 Angular velocity of UGV whilst navigation in scenario-IV using FIS.

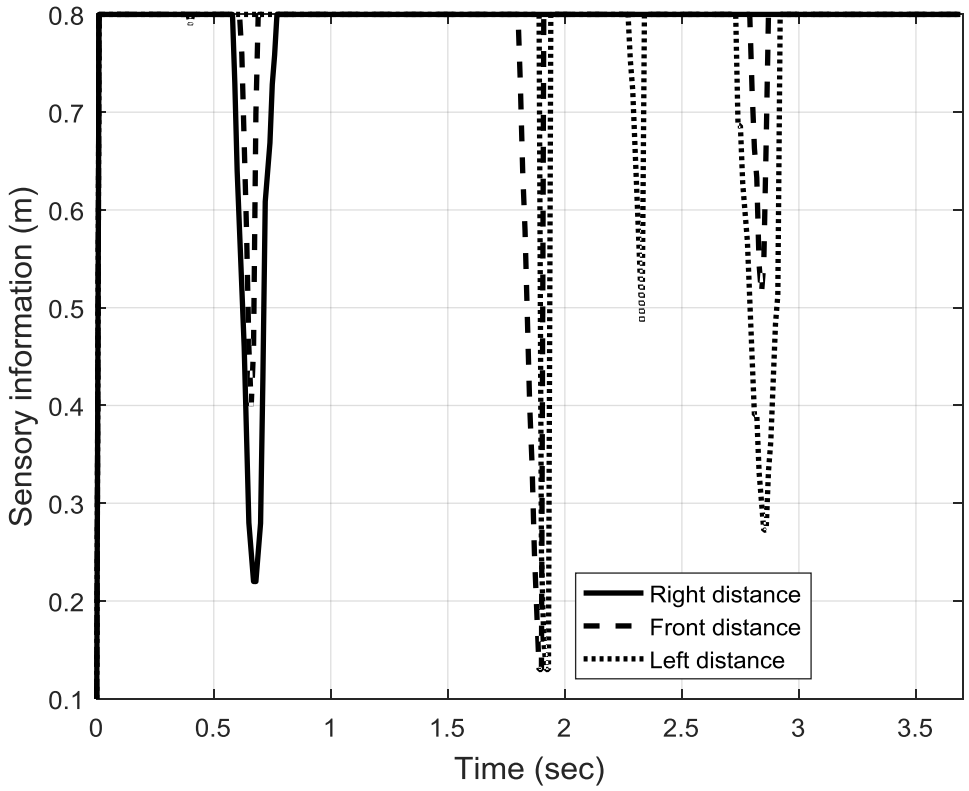



Fig. 6.51 Sensory information of three sensors of UGV in scenario-IV using FIS.

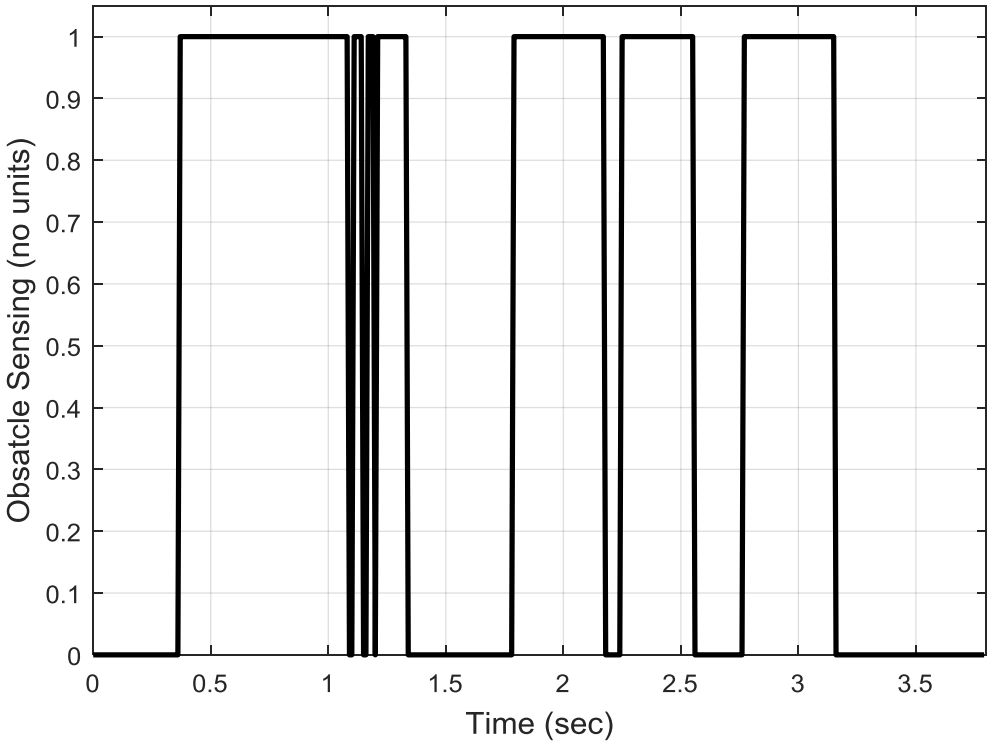



Fig. 6.52 Obstacle sensing indicator whilst navigation in scenario-IV using FIS.

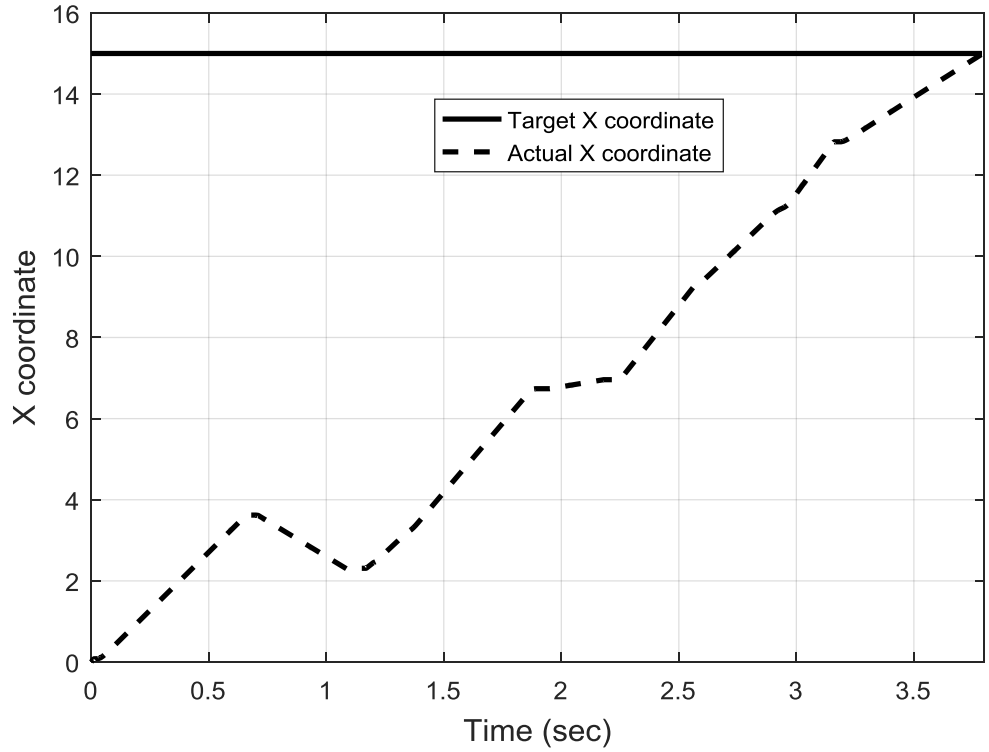



Fig. 6.53 X-coordinates of UGV whilst navigation in scenario-IV using FIS.

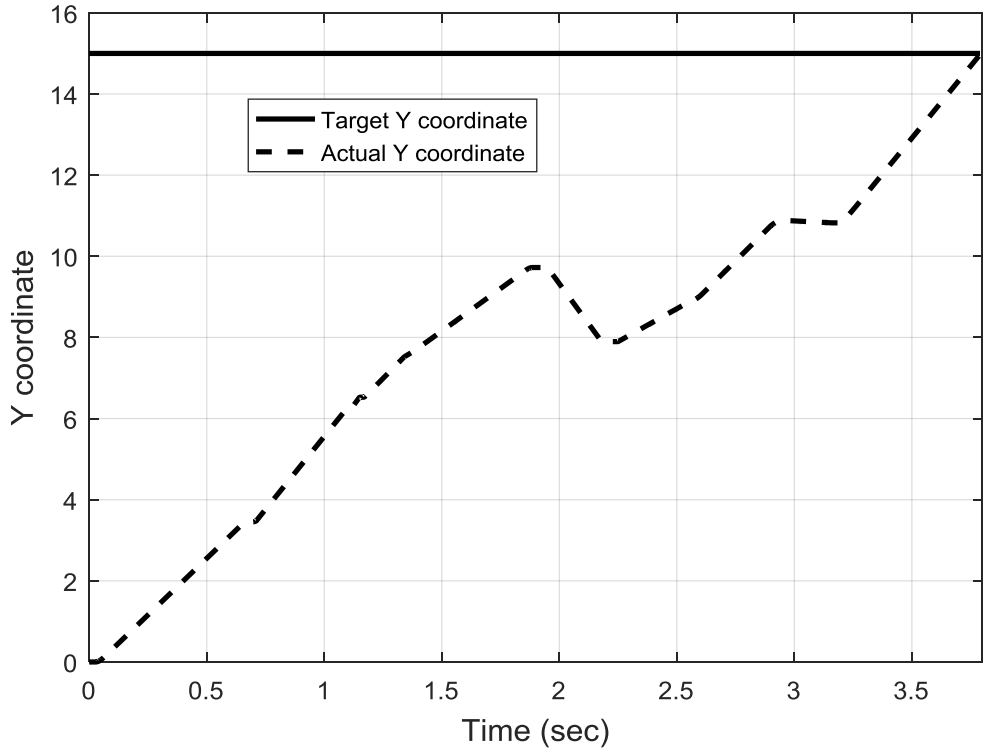



Fig. 6.54 Y-coordinates of UGV whilst navigation in scenario-IV using FIS.

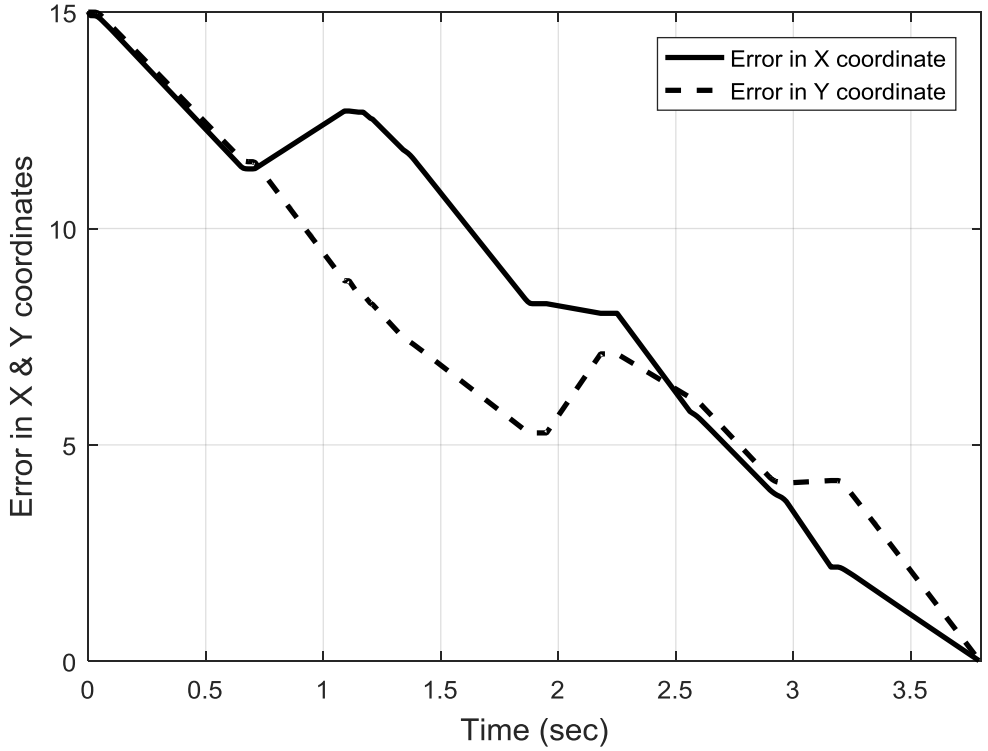



Fig. 6.55 Tracking error in X and Y coordinates of UGV in scenario-IV using FIS.

## 6.7 Comparisons with the related work

In order to validate the effectiveness of the proposed methodology, comparisons are made between the proposed approach and an approach that introduced in the state of the art by (Cherni et al., 2016). The comparisons are conducted based on two different scenarios. The first scenario is considered with three moving obstacles. The starting point is set at (0, 0) and the target point is placed at (8, 9). The three obstacles are positioned at different locations and orientations as demonstrated in Fig. 6.56. It is observable that the obtained path using our approach is more optimal in comparison to the state of the art method. The major advantages of our control system are the capability of reaching the destination in much shorter path whilst avoiding obstacles and minimising the elapsing time. In addition, our proposed system encompasses a smoothness of controllers' commands and its extensibility to compound scenarios.

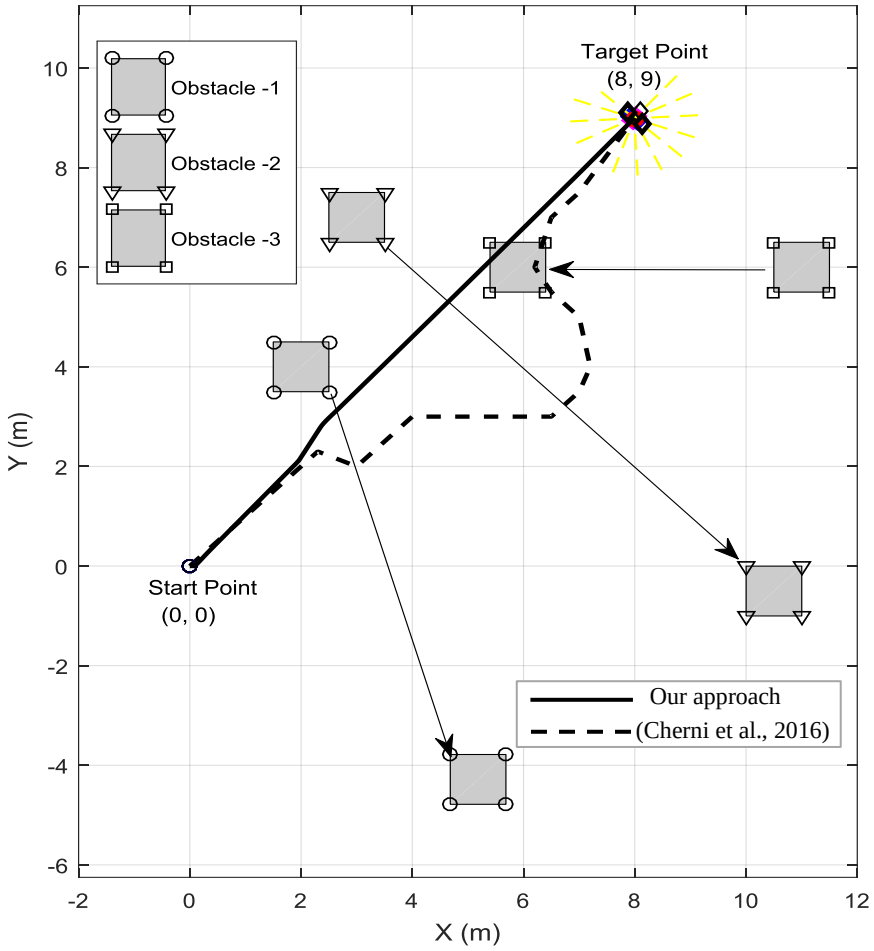



Fig. 6.56 First comparison of proposed FIS with Ref.(Cherni et al., 2016).

The other comparison is conducted based on a scenario that has obstacles traverse into constant and horizontal orientations. This leads to a different topology because new coordinates of the obstacles are occupied as shown in Fig. 6.57. In addition, the target point has been changed and it is placed at (12, 12). It is noticeable that the new generated path is quite different from Fig. 6.56 above, in response to the actual motion of the unmanned ground vehicle whilst navigating. That verifies the performance of the UGV in different circumstances that is capable of an instantaneous localisation and an effective adaptation to response to new conditions at any time within its workspace. It is noticeable that there is a spike close to the target point. In fact, this is made when the UGV approached to the first obstacle to avoid collision with it.

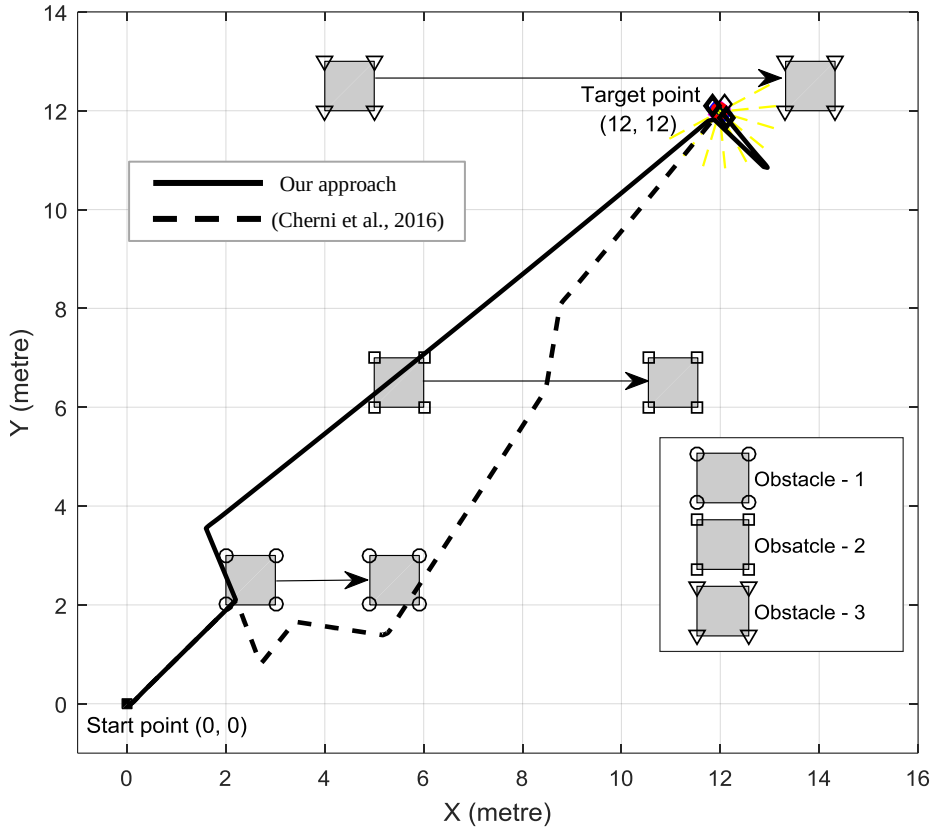



Fig. 6.57 Second comparison of proposed FIS with Ref.(Cherni et al., 2016).

## 6.8 Chapter Summary

In this chapter, an effective fuzzy inference system is presented for guiding an unmanned ground vehicle in a cluttered dynamic environment. The FIS makes the UGV move to a pose in a workspace whilst avoiding moving obstacles in the vehicle's path. The introduced FIS system consists of two controllers based on reactive behaviour. Firstly, the target reaching controller is to ensure that the vehicle reaches its destination point. In this controller, the input is the angle difference between vehicle's heading and target angle. Secondly, it is to use the obstacle avoidance controller, in this controller, the vehicle receives sensory information as inputs form three sensors placed on front, right and left of the vehicle's platform. The outputs of these two controllers are right and left angular velocities to guide the vehicle during its navigation.

The proposed FIS in random dynamic environments have been demonstrated in four scenarios. Firstly, five moving obstacles move at constant velocities. Secondly, the obstacles move in different velocities. Thirdly, the sizes of obstacles are varied to obtain a realistic real-world situation. Finally, static and dynamic obstacles are combined to demonstrated a diverse scenario. In all constructed environments, the UGV is proved that it has the capability of avoiding obstacles safely and reaches the destination with feasible and smooth path between the starting and the target points. Hence, FIS is evident to be a satisfactory control methodology for the UGV. It exhibits an intelligent behaviour whilst confronting of an uncertainty in the presence of the static and dynamic obstacles. It has been established that the proposed methodology is capable of effectively guiding the UGV to traverse from a starting position to a desired goal with the optimum and the shortest path based on the target reaching algorithm.

Comparisons are established to determine the privileges of the proposed design comparing to the state of the art. It has been confirmed that an elapsed time and an optimum path are obtained based on the obstacle avoidance and target reaching algorithms.

# Chapter 7

# Navigation of UGV Based on Adaptive Neuro-Fuzzy Inference System

## 7.1 Introduction

Fuzzy inference systems (FIS) have understandable structures about how decisions are made. However, they are not able to obtain the required inference rules automatically. Notwithstanding, the FIS responses within a fast period in a control process. Designing an optimal structure requires inevitable efforts to decide the inference rules between inputs and outputs upon linguistic variables. In addition, it needs to determine the effective inference rule numbers. On the other hand, the artificial neural networks are utilised widely in different applications and they demonstrate an adequate performance. However, they are not sufficient in explaining how their control decisions are reached. As a result, such problems reveal a strong need for creating an intelligent hybrid system that combines two or more techniques properly to overcome the limitations of the individual techniques. Therefore, the adaptive neuro-fuzzy inference system (ANFIS) has been introduced instead of the fuzzy inference systems and neural networks.

A neuro-fuzzy model brings together the linguistic representation of a fuzzy system with the learning ability of artificial neural networks. Hence, such a combination exhibits the advantages of both approaches to tackle many engineering problems. Consequently, by using the ANFIS model, the disadvantages of the fuzzy logic systems and the artificial neural networks will vanish. The neuro-adaptive training techniques provide a method for the fuzzy modelling procedure to train information about provided datasets. Therefore, it provides the fuzzy inference system the capability to compute the parameters of the membership and rules effectively that minimise the error rate between the actual and predicted outputs. More benefits are provided by ANFIS model such as the learning and predicting efficiency based on the well-selected datasets.

The ANFIS model is one of Neuro-Fuzzy systems that allows the fuzzy systems to learn the parameters by using either an adaptive back-propagation learning algorithm or a hybrid-learning algorithm that combines back propagation and a mean square estimator. The ANFIS architecture is based on a Sugeno fuzzy model. This model is different from a Mamdani fuzzy

model in terms of fuzzy rules are applied. Thus, their aggregation and defuzzification procedures differ accordingly. The rule implementation based on the Sugeno fuzzy model is defined as a linear combination of input variables. The corresponding final output is simply the weighted average of each rule's output. A Sugeno fuzzy model consisting of two input variables x and y, for example, one output variable f will lead to two fuzzy rules:

$$\text{Rule 1: If } x \text{ is } A_1, y \text{ is } B_1 \text{ then } f1 = p_1x + q_1y + r_1$$

$$\text{Rule 2: If } x \text{ is } A_2, \text{ y is } B_2 \text{ then f2} = p_2x + q_2y + r_2$$

where $p_i$, $q_i$, and $r_i$ are the consequent parameters of $i^{th}$ rule. $A_i$, $B_i$ and $C_i$ are the linguistic labels of inputs. The architecture of the ANFIS will be discussed in detail in the following section

Following a brief review of ANFIS's advantages and applications to various problems in industrial automation. In this chapter, we aim to develop ANFIS models for solving the navigation problem of unmanned ground vehicles in industrial environments that are filled with unknown obstacles. It means the UGV does not have a prior knowledge about the places of obstacles. However, it is assumed that obstacles are invariant which they do not change their position with the time.

## 7.2 Chapter Organisation

The chapter is organised as follows: In the following section, a brief overview of the architecture of an adaptive neuro-fuzzy inference system is introduced. Section 7.4 is dedicated to design ANFIS controllers for navigation platform. The simulation results are conducted and illustrated in Section 7.5. A comparison with the related work is given in Section 7.6. Chapter summary is described in Section 7.7.

## 7.3 Architecture of Adaptive Neuro-Fuzzy Inference System

The architecture of an adaptive neuro-fuzzy Inference system (ANFIS), consists of a fuzzy inference system and a neural network with given input and output data pairs. The ANFIS technique is a self-tuning and an adaptive hybrid controller that uses learning algorithms to tune its performance. In other words, it provides the fuzzy inference system the capability to adapt membership function parameters that it allows an associated fuzzy inference system to track given input and output data parameters of ANFIS model. In order to process a fuzzy rule by neural networks, it is necessary to modify a standard neural network structure accordingly. Fig. 7.1 demonstrates the architecture model of ANFIS. This model is called a first-order

Takagi-Sugeno-fuzzy model (Jang, 1993). For simplicity, it is assumed that the ANFIS model has two inputs $k_1$ and $k_2$, and one output f.

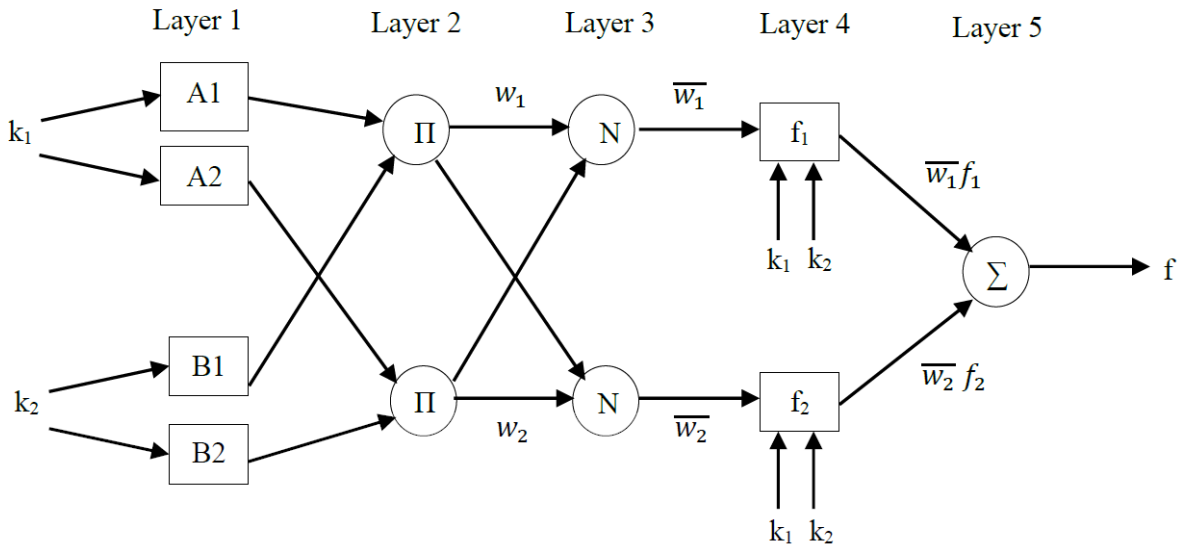



Fig. 7.1 Architecture of an adaptive neuro-fuzzy inference system.

This system can be broken down into five layers:

**Layer 1**: Every node 'i' in this layer is an adaptive node with a node function

$$O_{1,i} = \mu_{A\_i}(x), for\ i = 1,2 \quad (7.1)$$

$$O_{1,i} = \mu_{B_{i-2}}(y), for\ i = 3,4 \quad (7.2)$$

where $k_1$ (or $k_2$) is the input to node i and $A_i$ (or $B_{i-2}$) is a linguistic label (such as "near" or "far") associated with a particular node. In other words, $O_{1,i}$ is the membership grade of a fuzzy set A( = $A_1$ , $A_2$ , $B_1$ or $B_2$ ) and it specifies the degree to which the given input $k_1$ (or $k_2$) satisfies the quantifier A. The membership function for A is assumed be triangular shaped membership function:

$$\mu_A(x) = max\left[min\left\{\frac{x-a_i}{b_i-a_i}, \frac{c_i-x}{c_i-b_i}\right\}, 0\right] \quad (7.3)$$

where $\{a_i, b_i, c_i\}$ is the parameter set. As the values of these parameters change, the bell-shaped function varies accordingly, thus exhibiting various forms of membership function for the fuzzy set A. Parameters in this layer are referred to as premise parameters.

**Layer 2:** Every node in this layer is a fixed node labelled Π, whose output is the product of all the incoming signals:

$$O_{2,i} = w_i = \mu_{A\_i}(x)\mu_{B_i}(y), i = 1,2 \tag{7.4}$$

Each node output represents the firing strength of a rule. In general, any other T-norm operators that perform a fuzzy AND can be used as the node function in this layer.

**Layer 3:** Every node in this layer is a fixed node labelled N. The $i^{th}$ node calculates the ratio of the $i^{th}$ rule's firing strength to the sum of all of the rules firing strengths:

$$O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2}\ i = 1,2 \tag{7.5}$$

The output of this layer is called the normalised firing strengths.

**Layer 4:** Every node *i* in this layer is an adaptive node with a node function:

$$O_{4,i} = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i), i = 1,2 \tag{7.6}$$

where $\overline{w_i}$ is the normalised firing strength from layer 3 and $\{p_i, q_i, r_i\}$ is the parameter set of this node. Parameters in this layer are referred to as consequent parameters.

**Layer 5:** The single node in this layer is a fixed node labelled ∑, which computes the overall output, f, as the summation of all incoming signals:

$$f = O_{5,1} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}\ i = 1,2 \tag{7.7}$$

The first and fourth layers are adaptive layers in the ANFIS architecture. The modifiable parameters are called premise parameters in the first layer and consequent parameters in the fourth layer. The task of learning is to tune all modifiable parameters to make the ANFIS match the training data. Trainable parameters of ANFIS, *i.e.*, premise parameters and consequent parameters $\{a_i, b_i, c_i\}$ and $\{p_i, q_i, r_i\}$ are adjusted to make the ANFIS output match the training data. The utilised training algorithm is called a hybrid learning algorithm. It combines the least square method and gradient descent method. The hybrid learning algorithm is composed of two processes i.e. a forward pass and a backward pass. The least squares method (forward pass) is used to optimize the consequent parameters with the premise parameters fixed. Once the optimal consequent parameters are found, the backward pass starts immediately. The gradient descent method (backward pass) is used to adjust optimally the premise parameters corresponding to the fuzzy sets in the input domain. The output of the ANFIS is calculated by

employing the consequent parameters found in the forward pass. The output error is used to adapt the premise parameters by means of a standard back propagation algorithm. The performance of the ANFIS network is evaluated statistically based on a root mean square error (RMSE). The error rate is measured between the actual and predicted values of the ANFIS output by employing the following equation:

$$RMSE = \sqrt{\frac{1}{n}\sum (P_i - A_i)^2} \tag{7.8}$$

where $P_i$ and $A_i$ are respective predicted and actual yield for the $i^{th}$ iterations of data pairs and $n$ is the number of the points in the dataset.

## 7.4 Design of ANFIS Controllers for Navigation Platform

In this section, the proposed ANFIS controllers are discussed. Four ANFIS controllers have been designed to accomplish the navigation task; firstly, two ANFIS controllers for achieving the target reaching task; secondly, two ANFIS controllers for performing the obstacle avoidance mission.

All four controllers are combined through a switch block for choosing which controller will be activated. For instance, if there are no obstacles in the vehicle's path, the target-reaching controller will be activated. Otherwise, if the vehicle senses an obstacle, the obstacle avoidance controller is activated. The switching between these two controllers is decided according to the obstacle sensing signal, $O_S$, from the environmental model. This signal is generated in accordance to measured distances (front, right, and left) from sensory information. If the vehicle does not sense an obstacle in its path, this OS parameter indicates '0' if there is no an obstacle and '1' if the vehicle senses an obstacle near to its platform. Thus, the output of the switching block are the angular velocities of the left and right wheel of the unmanned ground vehicle. These velocities are provided in the vehicle's model to obtain the instantaneous vehicle's posture through the movement of the vehicle. Fig. 7.2 depicts the structure of the proposed ANFIS controllers. These controllers replace the fuzzy inference systems presented in the previous chapter. They are integrated with the UGV model and the introduced workspace environment.

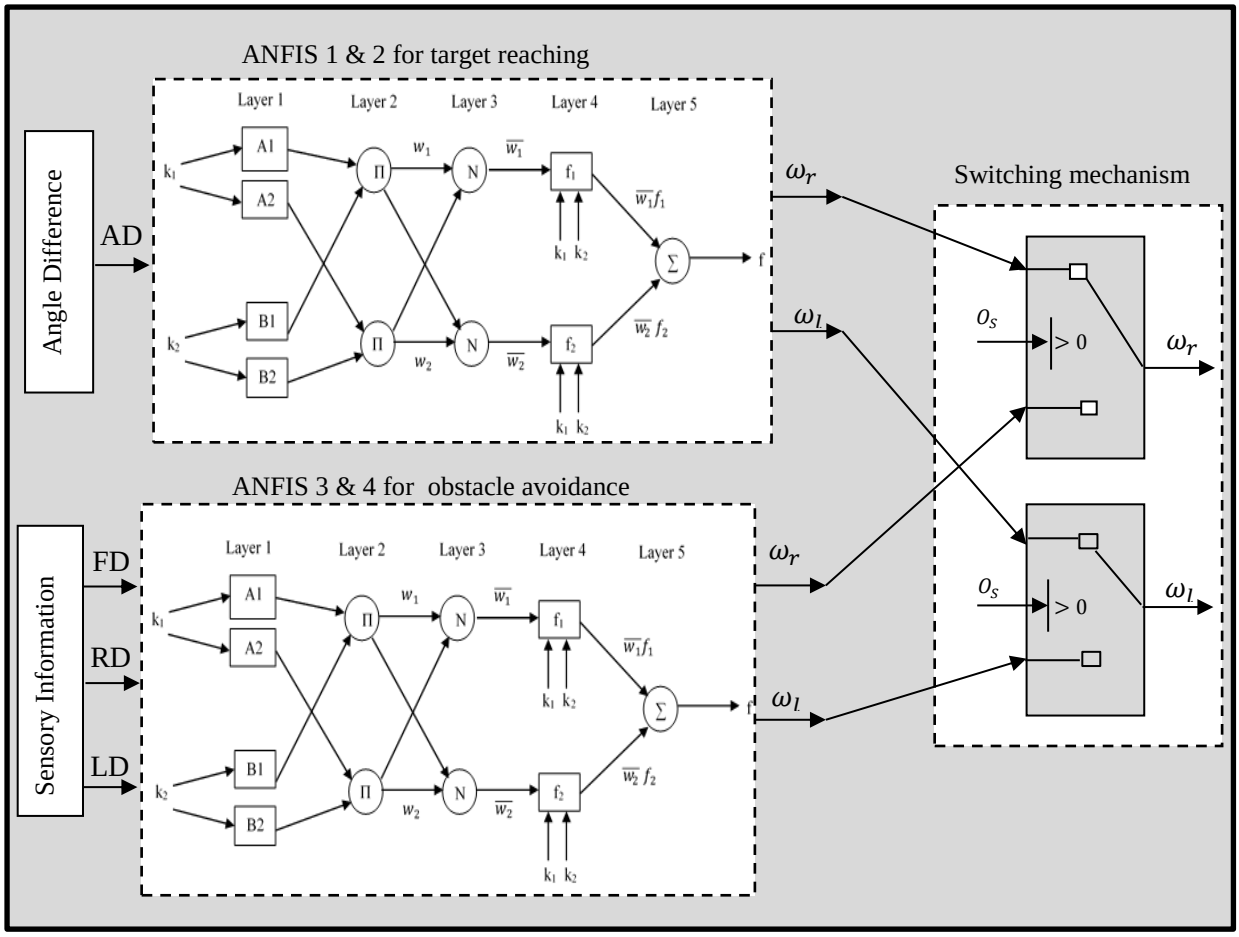



Fig. 7.2 Block diagram of proposed four ANFIS controllers.

### *7.4.1 Hybrid Training Algorithm*

The hybrid-training algorithm is used to modify the parameters of the ANFIS model as follows: The gradient descent method as in neural network can be applied to modify the premise parameters whilst a least square estimate method can be applied to adapt the consequent parameters. In the forward pass of the hybrid learning algorithm, functional signals go forward (under the condition that the premise parameters are fixed) until layer four and the consequent parameters are identified by the least square estimate. In the backward pass, the error rates propagate backward and the premise parameters are updated by using the back propagation algorithm that described in Chapter 5.

### *7.4.2 Target Reaching ANFIS Controller*

To achieve a target reaching ANFIS architecture, a single input multiple output (SIMO) ANFIS controller is needed that comprises of one input and two outputs. However, the MATLAB-SIMULINK package does support only a multiple input single output (MISO) or a

single input single output (SISO) ANFIS models. Hence, two ANFIS controllers are implemented to provide two outputs for driving the right and left angular velocities toward the target. Both ANFIS controllers share the same angle difference input. This angle difference represents a subtraction between the vehicle's heading and the target's point. The calculation of the angle difference is already given in Chapter 6. The output of the first ANFIS controller is the right angular velocity and for the second ANFIS controller is the left angular velocity. A group of datasets has been selected after filtering the recursion in the datasets for training both ANFIS controllers. The datasets have been chosen based on real values that are obtained from the fuzzy inference system implemented preciously. The training procedure adjusts membership parameters to obtain an optimal performance of the model. The learning information of the first and second ANFIS controllers for predicting the angular velocity of the left and the right wheels respectively are given in Table 7.1. The characteristics of the structure of ANFIS architecture are illustrated in Table 7.2.

The training results illustrate that the error rate for predicting the left and right angular velocities are '0.015 rad/s' and '0.0523 rad/s', respectively. An epoch number of 200 is chosen after based on many iterations for obtaining the minimum value of error for training the datasets. The relationship between the error rate and epoch number is demonstrated in Fig. 7.3 for both ANFIS 1 and 2. Ten membership functions have been specified in the angle difference input. The number of membership functions can be increased to minimise the error rate further. However, that will increase the complexity of the ANFIS architecture. Hence, the elapsed time will be increased and that would cause a delay in the response of the ANFIS model. MATLAB's ANFIS editor offers different types of membership functions (MF) including: triangular and trapezoidal. Correspondingly, we have chosen the triangular membership function after evaluating the other types as it has provided the best results.

Table 7.1 Learning information of the first and the second topology of ANFIS.

| ANFIS information | ANFIS 1 | ANFIS 2 |
|---|---|---|
| Number of nodes | 52 | 52 |
| Number of linear parameters | 12 | 12 |
| Number of nonlinear parameters | 36 | 36 |
| Total number of parameters | 48 | 48 |
| Number of training data pairs | 350 | 350 |
| Number of fuzzy rules | 12 | 12 |

Table 7.2 Characteristics of ANFIS 1 and 2 architecture.

| Item | Type of MF | | Number of MF | | Learning method | RMSE |
|---|---|---|---|---|---|---|
| | Input | Output | Input | Epoch | | |
| ANFIS 1 | Triangle | Linear | 10 | 200 | Hybrid | 0.0015 |
| ANFIS 2 | Triangle | Linear | 10 | 200 | Hybrid | 0.0523 |

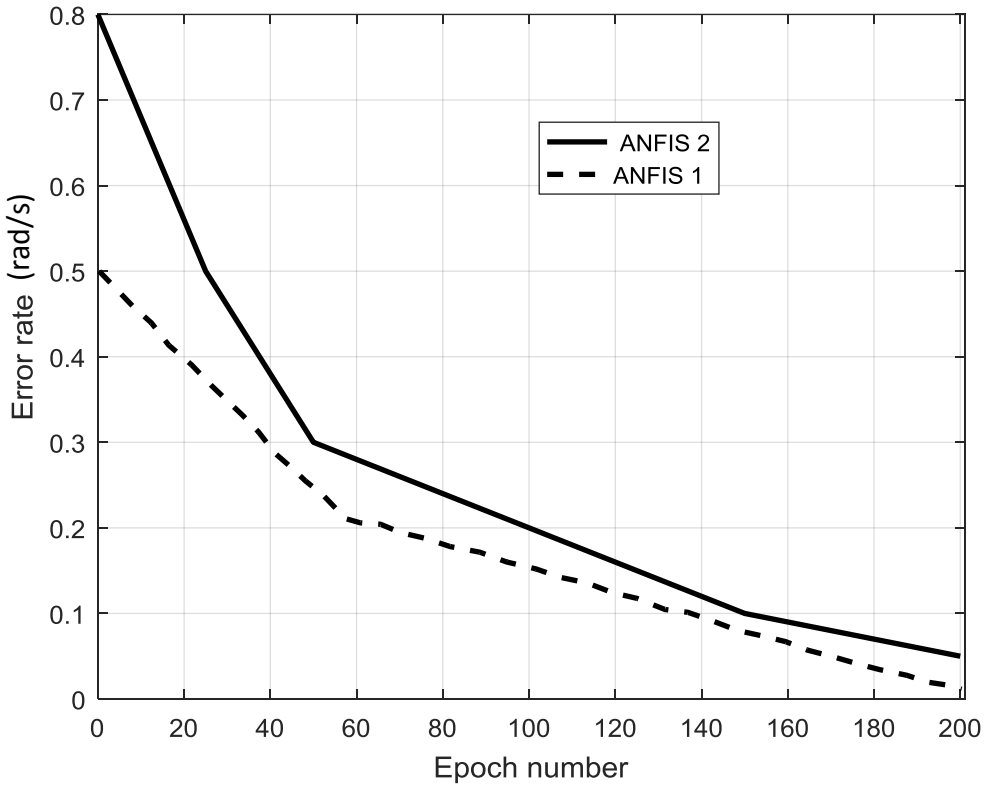



Fig. 7.3 Error rate for ANFIS controllers of target reaching.

### *7.4.3 Obstacle Avoidance ANFIS Controller*

The principal operation of an unmanned ground vehicle is to have a collision free path during the navigation. In this architecture, there are four inputs and two outputs. The inputs are the front, right and left distances, and the outputs are the right and left angular velocities. Similarly, because of the ANFIS architecture based on MATLAB software package provides just one output. Currently, our model is a multiple input multiple output (MIMO). Therefore, the model is divided into two MISO models. Hence, Both of MISO models are deployed to implement the third and fourth ANFIS controllers. These controllers are utilised to steer the vehicle's orientation when the vehicle becomes near obstacles.

For implementing the obstacle avoidance ANFIS controller, 35 pairs of training datasets are selected for the training purpose. The chosen training datasets are based on the most

dissimilar figures among a large volume of datasets. The training adjusts the membership parameters to implement the required model. The ANFIS learning information for predicting the angular velocity for the left and the right wheels are illustrated in Table 7.3. The characteristics of the structure of ANFIS architecture are described in Table 7.4. It is observable that the type of MF for which of three inputs is triangular and each of the inputs has five membership functions. Training result shows that the error rate between the predicting and actual angular velocity of ANFIS3 and ANFIS4 are '0.178' rad/s and '0.102 rad/s', respectively. The relationships between the error rate and epoch number of both ANFIS controllers are demonstrated in Fig. 7.4.

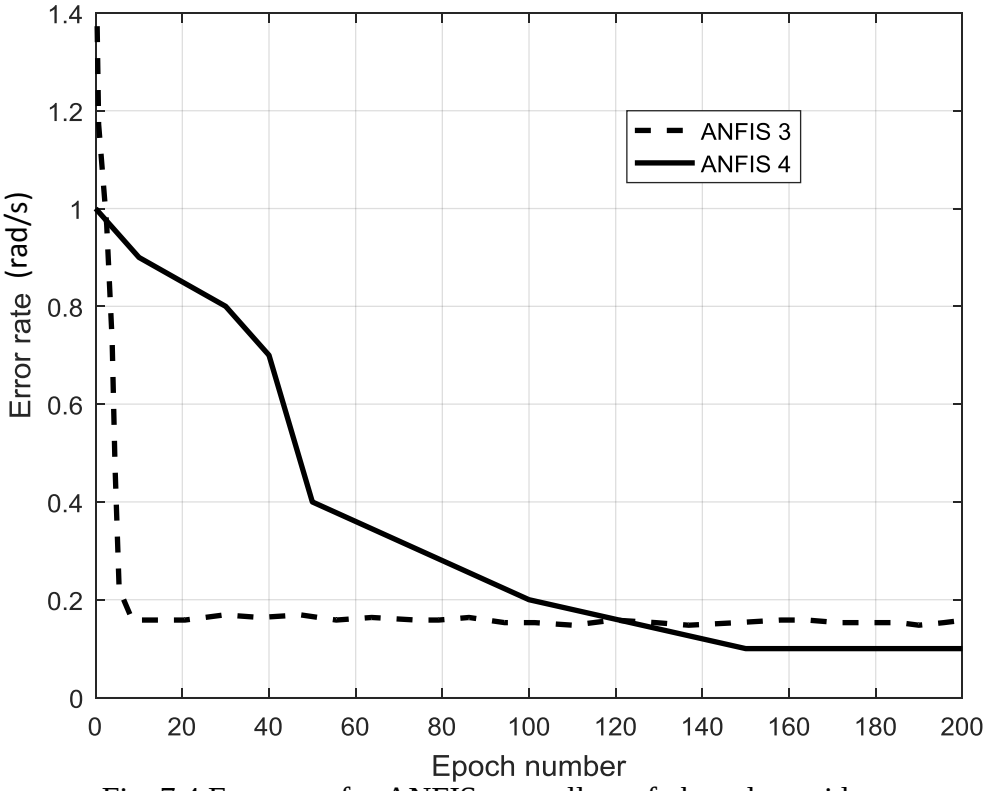



Fig. 7.4 Error rate for ANFIS controllers of obstacle avoidance.

Table 7.3 ANFIS information of the first and the second topology of ANFIS.

| ANFIS information | ANFIS 3 | ANFIS 4 |
|---|---|---|
| Number of nodes | 286 | 286 |
| Number of linear parameters | 125 | 125 |
| Number of nonlinear parameters | 45 | 45 |
| Total number of parameters | 170 | 170 |
| Number of training data pairs | 350 | 350 |
| Number of fuzzy rules | 125 | 125 |

Table 7.4 Characteristics of ANFIS 3 and 4 architecture.

| Item | Type of MF | | Number of MF | | Learning method | RMSE |
|---|---|---|---|---|---|---|
| | Input | Output | Input | Epoch | | |
| ANFIS 3 | Triangle | Linear | 5,5,5 | 200 | Hybrid | 0.178 |
| ANFIS 4 | Triangle | Linear | 5,5,5 | 200 | Hybrid | 0.102 |

## 7.5 Simulation Results

To validate the proposed ANFIS controllers, two case studies have been carried out and simulated based on the MATLAB-SIMULINK environment. Each case study is constructed from a variety of obstacles that are placed in different positions in the workspace. The initial and the destination points of both case studies are similar. However, the configurations and the sizes of obstacles are diverse. Nonetheless, the initial position of the UGV can be set arbitrarily in the workspace to reach any target. Both case studies are explained in detail as in the following subsections.

### *7.5.1 Case Study-I*

In this case, a workspace has been presented as demonstrated in Fig. 7.5 for the navigation platform filled with seven static obstacles. The workspace dimensions are fixed by four corner points having the coordinates (–2, –2), (18, –2), (18, 18), (–2, 18) to combine a two-dimensional grid. The dimensions of the obstacles described by their peripheral vertices and occupied spaces are given in Table 7.5. The Cartesian coordinates of the initial and target points are $P_o$ (0, 0) and $P_g$ (15, 15), respectively.

By running the implemented SIMULINK model, the UGV starts manoeuvring from the initial position towards to the destination position. It can be observed that, the UGV has avoided the surrounding obstacles successfully and safely. The decisions are made to change the vehicle's direction when it approaches any obstacle; this is illustrated in Fig.7.6 that demonstrates how the UGV alters its orientation to avoid the pertaining obstacle. The linear velocity of the UGV is introduced as shown in Fig. 7.7. The task of obstacle avoidance has been accomplished based on the activation of the ANFIS3 and ANFIS4, in this situation, the simulation results of angular velocity of both right and left wheels are shown in Fig. 7.8. Whereas, when there are no obstacles approaching the UGV's path, the ANIFS1 and ANFIS2 are activated to guide the UGV to reach its destination. Fig. 7.9 illustrates the behaviour of the left and right angular velocity when target-reaching controllers are activated. Accordingly, the

final left and right angular velocities after the stage of the switching mechanism are shown in Fig. 7.10. It demonstrates the reactions of the left and right wheels that are combined based to the obstacle avoidance and target-reaching controllers.

Table 7.5 Obstacles in the workspace grid of Case Study-I.

| **Obstacle No.** | **Peripheral Vertices Coordinates** |
|---|---|
| 1 | (8, 4), (8, 5.5), (9.5, 5.5), (9.5, 4) |
| 2 | (8, 14), (8, 15.5), (9.5, 15.5), (9.5, 14) |
| 3 | (10, 8), (10, 9.5), (11.5, 9.5), (11.5, 8) |
| 4 | (3.5, 12.5), (3.5, 14), (5, 14), (5, 12.5) |
| 5 | (2, 2), (2, 3.5), (3.5, 3.5), (3.5, 2) |
| 6 | (6, 7), (6, 8.5), (7.5, 8.5), (7.5, 7) |
| 7 | (12, 12), (12, 13.5), (13.5, 13.5), (13.5, 12) |

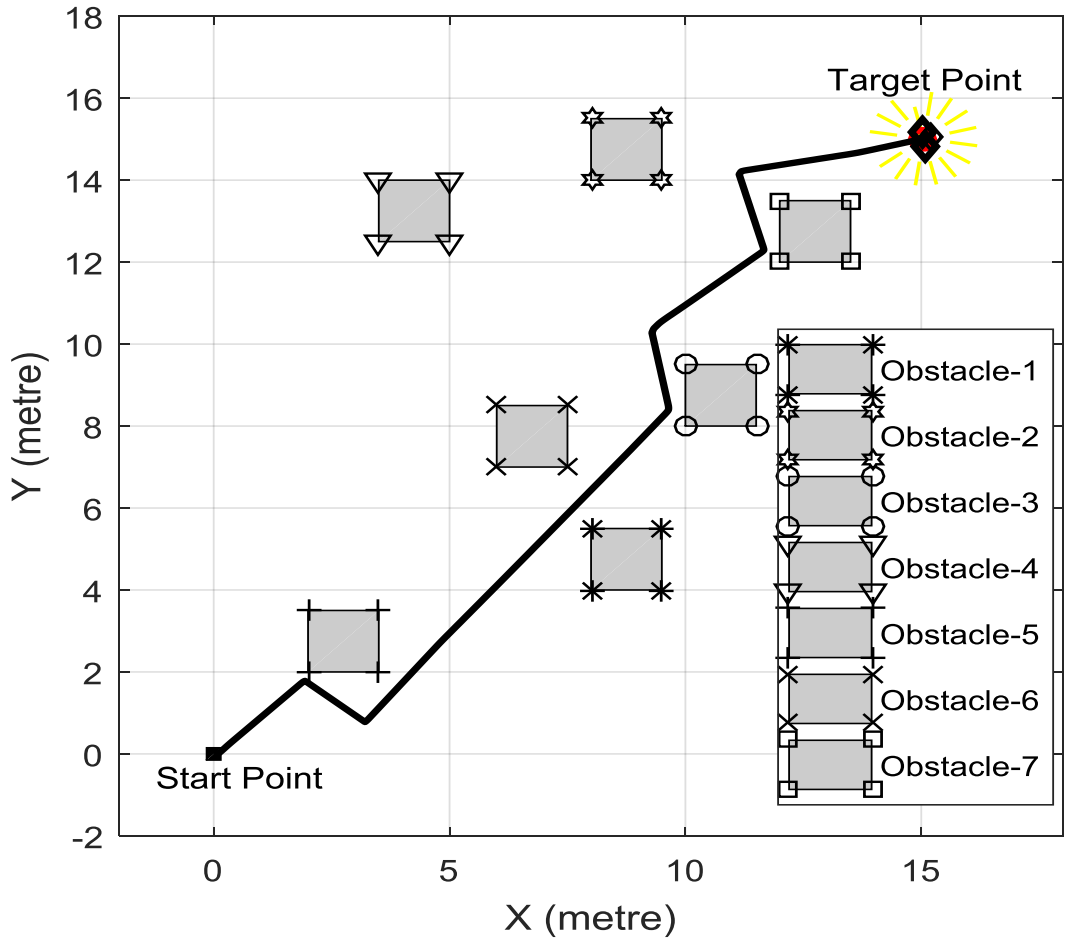



Fig. 7.5 Navigation platform and topology for case study-I using ANFIS.

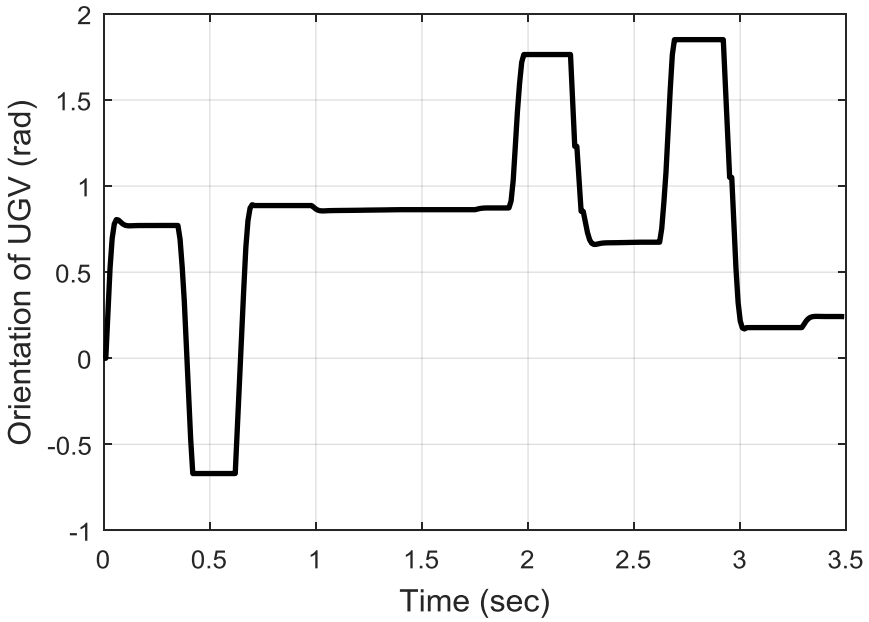



Fig. 7.6 Orientation of UGV whilst navigation in case study-I using ANFIS.

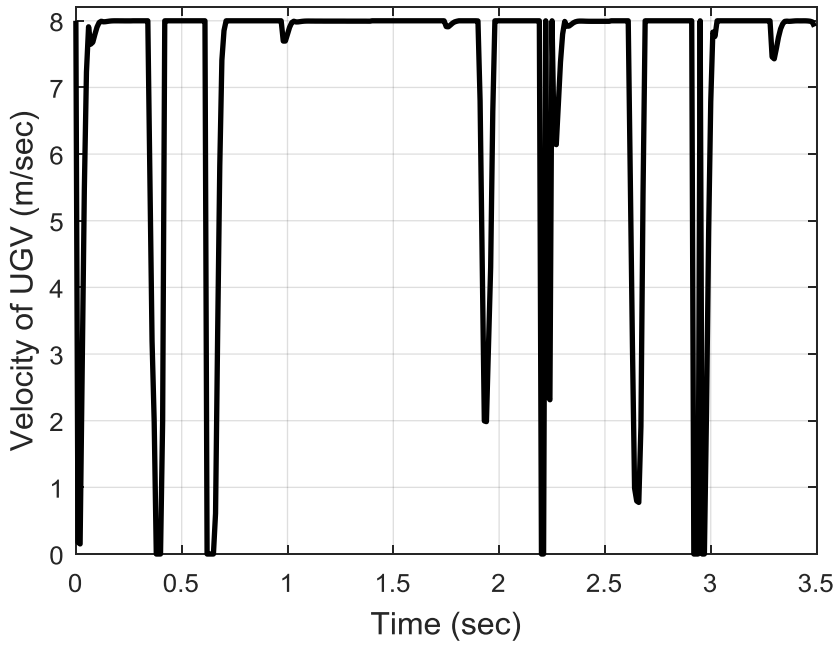



Fig. 7.7 Linear velocity for UGV in case study-I using ANFIS.

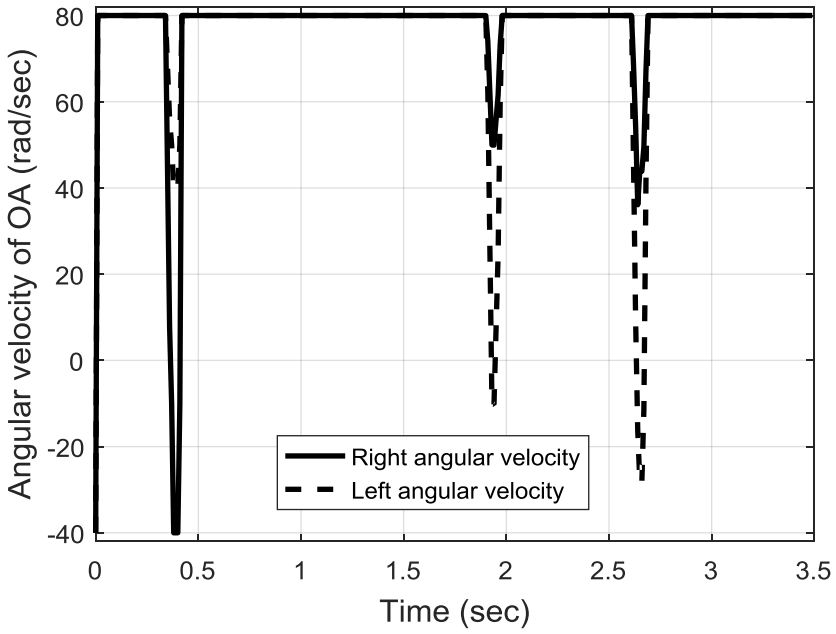



Fig. 7.8 Angular velocity for obstacle avoidance of ANFIS in case study-I.

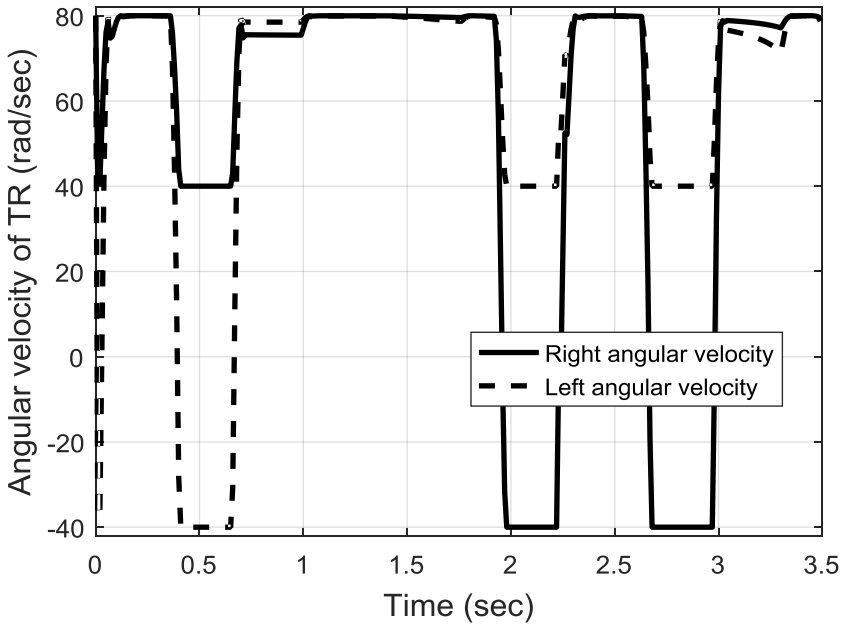



Fig. 7.9 Angular velocity for target reaching of ANFIS in case study-I.

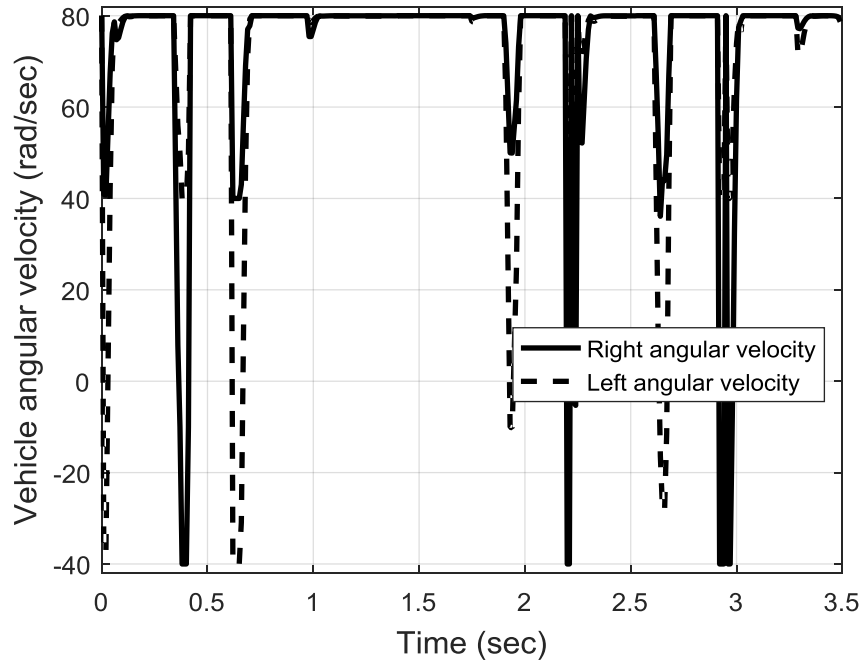



Fig. 7.10 Linear velocity of UGV whilst navigation in case study-I using ANFIS.

The action of the sensory information in this case study is demonstrated in Fig. 7.11. It is noticeable that the three sensors have responded effectively into the main regions that the collision avoidance is needed.

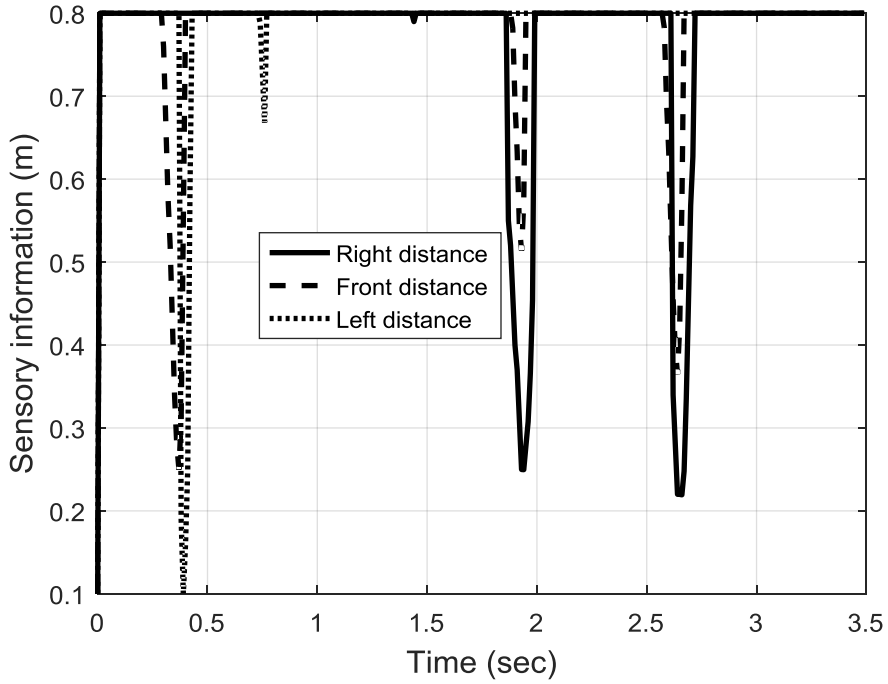



Fig. 7.11 Sensory information of three sensors of UGV in case study-I using ANFIS.

The Y and Y coordinates of the UGV motion are introduced in Fig. 7.12 and Fig.7.13, respectively. Both coordinates are conducted instantaneously from the start point until they reach the destination. The individual coordinates of each axis are positioned based on the posture of the UGV whilst manoeuvring. The trajectory tracking errors between the actual and the desired coordinates of X & Y axes are shown in Fig. 7.14.

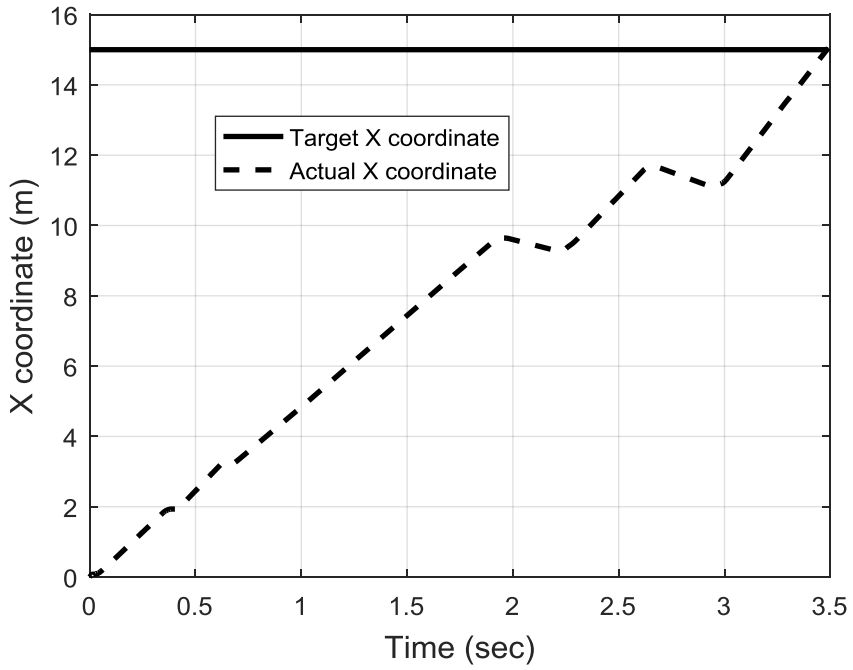



Fig. 7.12 X-coordinate of UGV whilst navigation in case study-I using ANFIS.

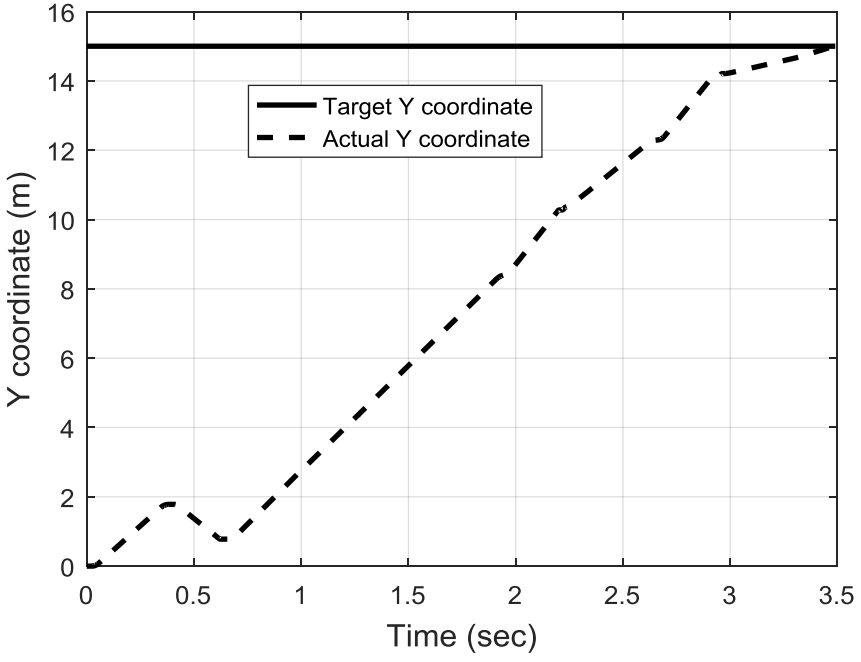



Fig. 7.13 Y-coordinate of UGV whilst navigation in case study-I using ANFIS.

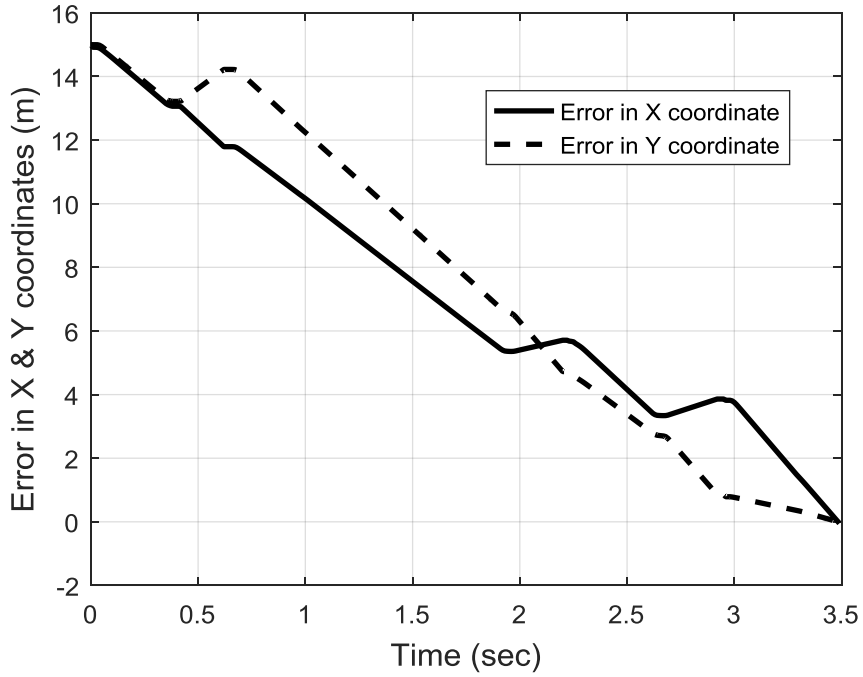



Fig. 7.14 Tracking error in X and Y coordinates of UGV in case study-I using ANFIS.

### *7.5.2 Case study-II*

In real world scenarios, obstacles would appear in different shapes and sizes based on unstructured environments. Thus, this case study is established based on an environment that is filled multiple varied obstacles. The complexity of the environment is considered by increasing the number of obstacles. It is expected that the frequency of the switching between the obstacle avoidance ANFIS controllers and the target reaching ANFIS controllers will be increased in case of increasing the number of obstructing obstacles. Consequently, the response might be more challenging and the UGV might lose the stability in a frequent switching. Hence, it is required to validate that the performance of the platform is adaptable into complicated situations and is capable to reach its destination and avoid obstacles in a highly various environment. The dimensions of the obstacles are described by their peripheral vertices, which are given in Table 7.6.

Table 7.6 Obstacles in the workspace grid of Case Study-II.

| **Obstacle No.** | **Peripheral Vertices Coordinates** | **Shape Type** |
|---|---|---|
| 1 | (3.5, 3.5), (5, 5), (2.5, 4), (4, 2.5) | Square |
| 2 | (8, 0), (9, 1.5), (10, 0) | Triangle |
| 3 | Centre (9, 4) and Radius = 0.75m | Circle |
| 4 | (12, 1), (12, 4), (14, 4), (14, 1) | Rectangle |
| 5 | (0, 6), (0, 8), (2, 8), (2, 6) | Square |
| 6 | (3.5, 3.5), (5, 5), (15.5, 4) | Triangle |
| 7 | (14.5, 8), (16, 8), (15.5, 6), (14, 6) | parallelogram |
| 8 | Centre (10, 10) and Radius = 1m | Circle |
| 9 | Centre (3, 13) and Radius = 0.75m | Circle |
| 10 | (12, 12), (12.5, 5), (13.5, 13), (14, 12) | Trapezoid |
| 11 | (8, 14), (8, 15.5), (9.5, 15.5), (9.5, 14) | Square |

The number of obstacles has been increased to eleven to make a more complex environment. The simulation results demonstrate that the trained obstacle avoidance ANFIS controllers have made the UGV traverses all the obstacles in its path by making decisions to change the vehicle's headings when it approaches an obstacle. Decisions are made upon the controllers' inputs to manipulate the left and right angular velocities. A new feasible trajectory is generated by the UGV's movement after roving around the obstacles as shown in Fig. 7.15. Accordingly, the orientation of the UGV is obtained as illustrated in Fig. 7.16. It demonstrates that the UGV has made several successful headings on both clockwise and counterclockwise. The linear velocity of the UGV is presented in Fig. 7.17. The right and left angular velocities of the target reaching and obstacle avoidance based on the conducted ANFIS controllers are demonstrated in Fig. 7.18 and Fig. 7.19, respectively. As a result, the right and left angular velocities of the target reaching and obstacle avoidance are combined through the switching mechanism to attain the required angular velocities for the left and right wheels. The ultimate left and right angular velocities are shown in Fig. 7.20.

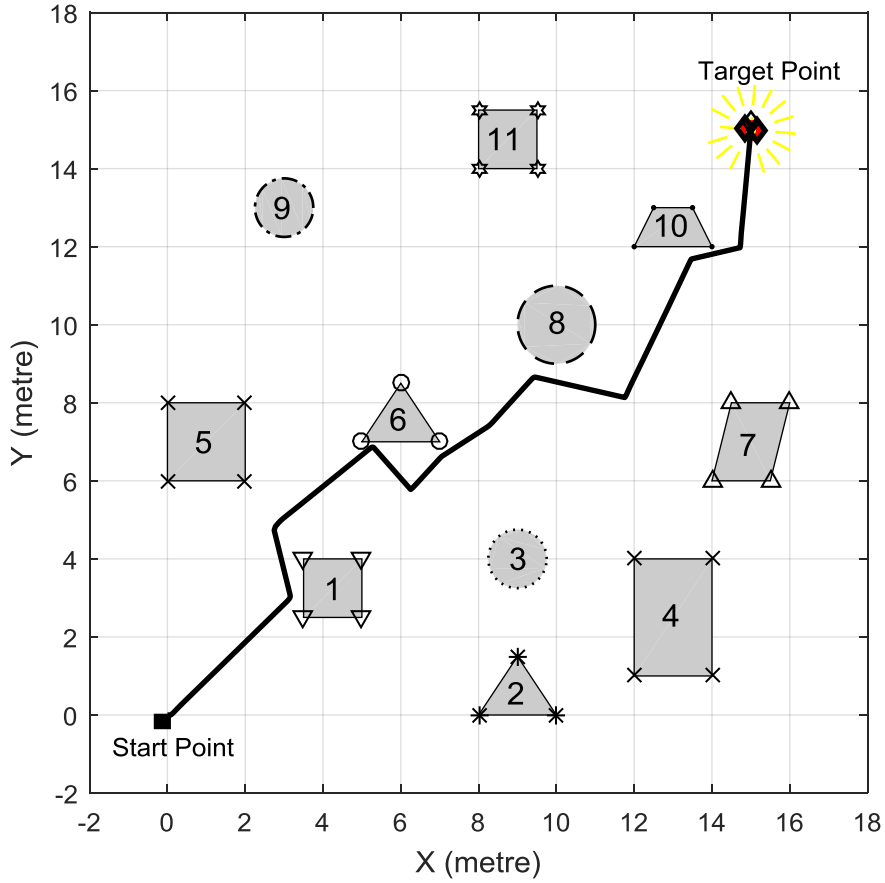



Fig. 7.15 Navigation platform and topology for case study-II using ANFIS.

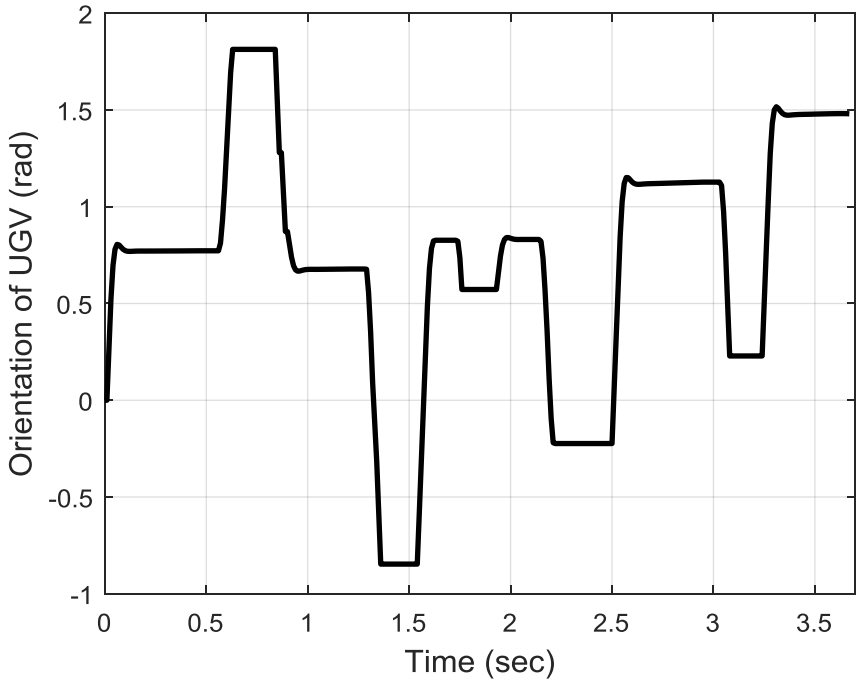



Fig. 7.16 Orientation of UGV whilst navigation in case study-II using ANFIS.

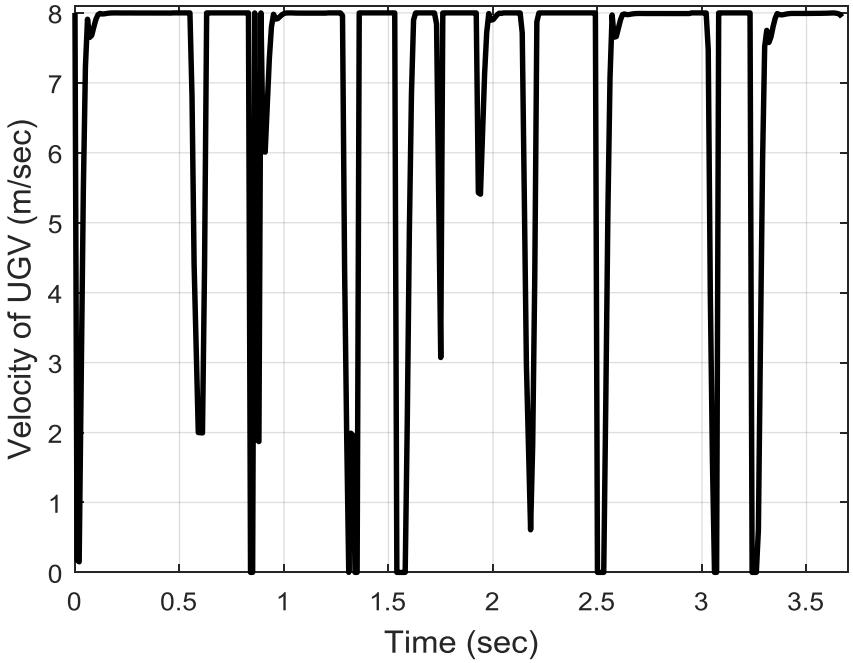



Fig. 7.17 Linear velocity for UGV in case study-II using ANFIS.

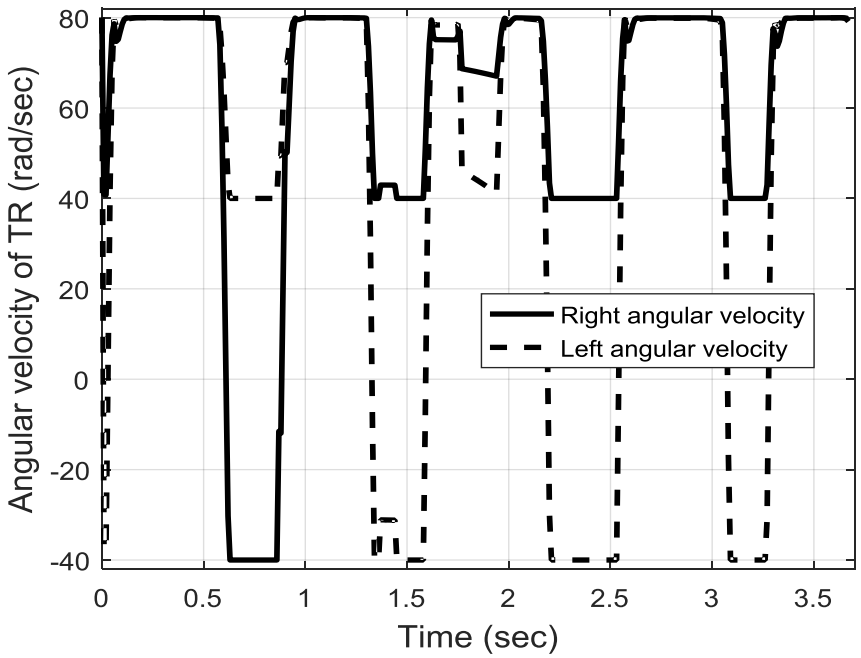



Fig. 7.18 Angular velocity for target reaching of ANFIS in case study-II.

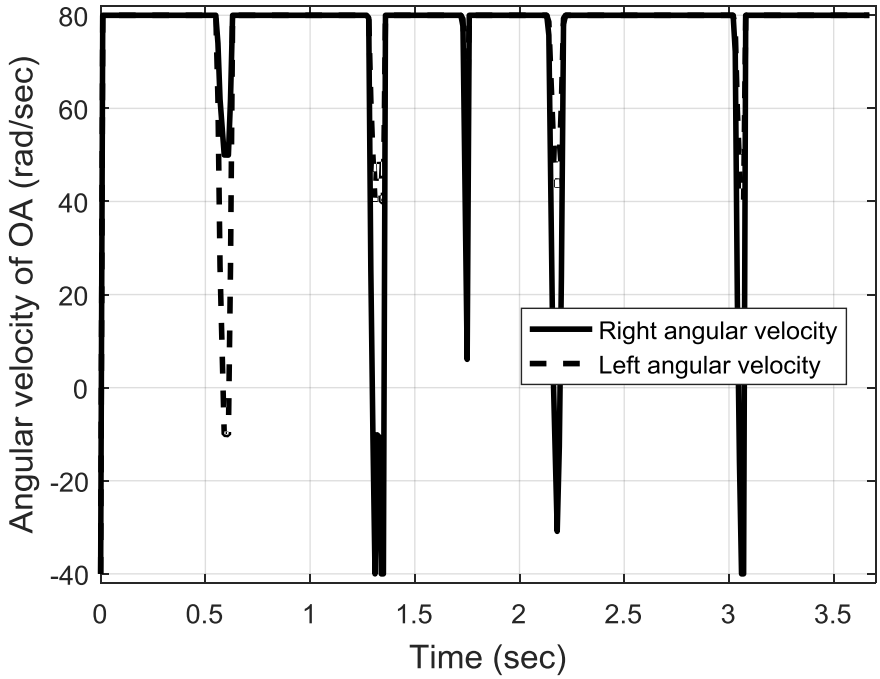



Fig. 7.19 Angular velocity for obstacle avoidance of ANFIS in case study-II.

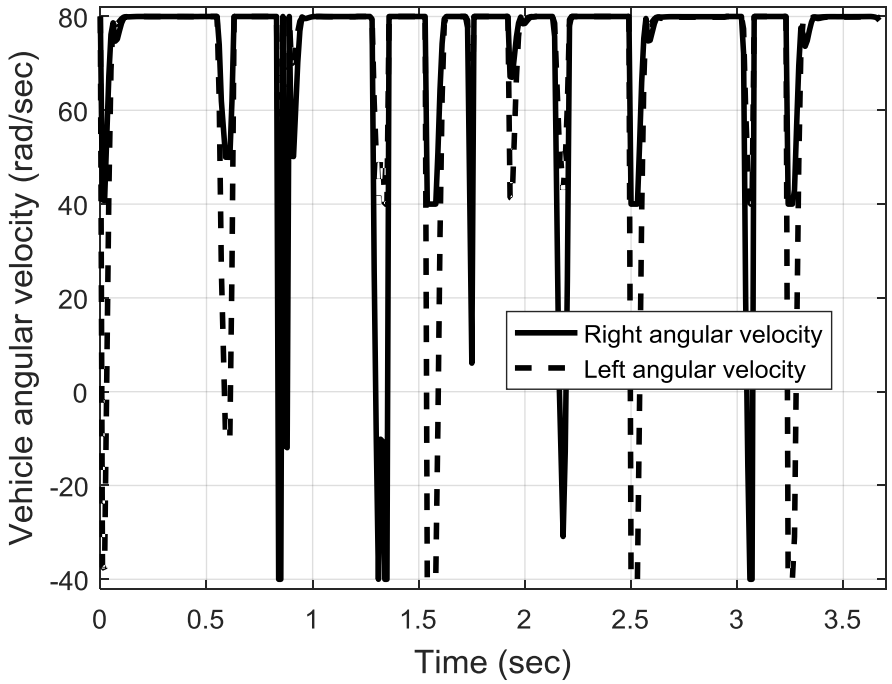



Fig. 7.20 Linear velocity of UGV whilst navigation in case study-II using ANFIS.

The sensor information given in Fig. 7.21 demonstrates that there are six significant responses occurring in reacting for approaching a hindrance. In addition, it is apparent that not all three sensors respond likewise. Instead, sensors who are obstructed by an obstacle will react accordingly when the UGV approaches the obstacle. As aforementioned in the previous case study, the motion coordinates of X and Y-axes are obtained as illustrated in Fig. 7.22 and Fig. 7.23, respectively. Similarly, the error rate between the actual and the destination coordinates is introduced in Fig. 7.24. It is evident that the UGV has passed all surrounding obstacles effectively and successfully despite the environment complexity. Moreover, the simulation results have confirmed that the UGV has reached its required destination. It is observable that, when the UGV has commenced its motion, the error rate has reached the maximum. Nonetheless, when the UGV approaches the coordinates of the destination, the error rate has been decreased constantly until it becomes zero. When the number of obstacles is increased and the sizes are varied, the obstacle avoidance ANFIS controller has been activated frequently in order to avoid obstacles. When the avoiding has been accomplished, the UGV has switched to the target reaching ANFIS controller based on the provided indicated sensing signal. In both considered case studies, the UGV has been capable of avoiding obstacles and reaching the target feasibly.

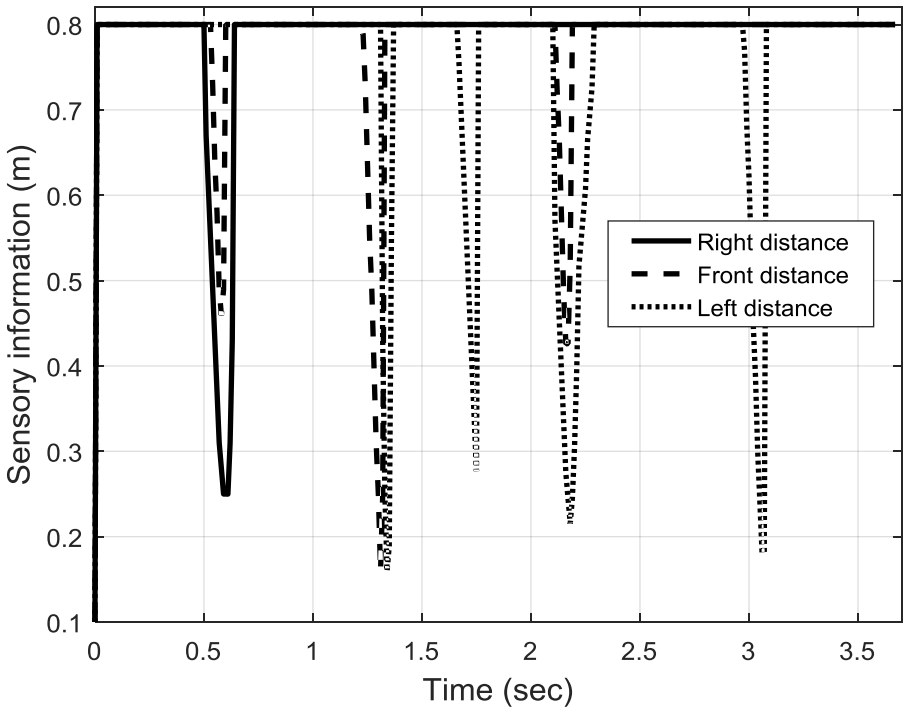



Fig. 7.21 Sensory information of three sensors of UGV in case study-II using ANFIS.

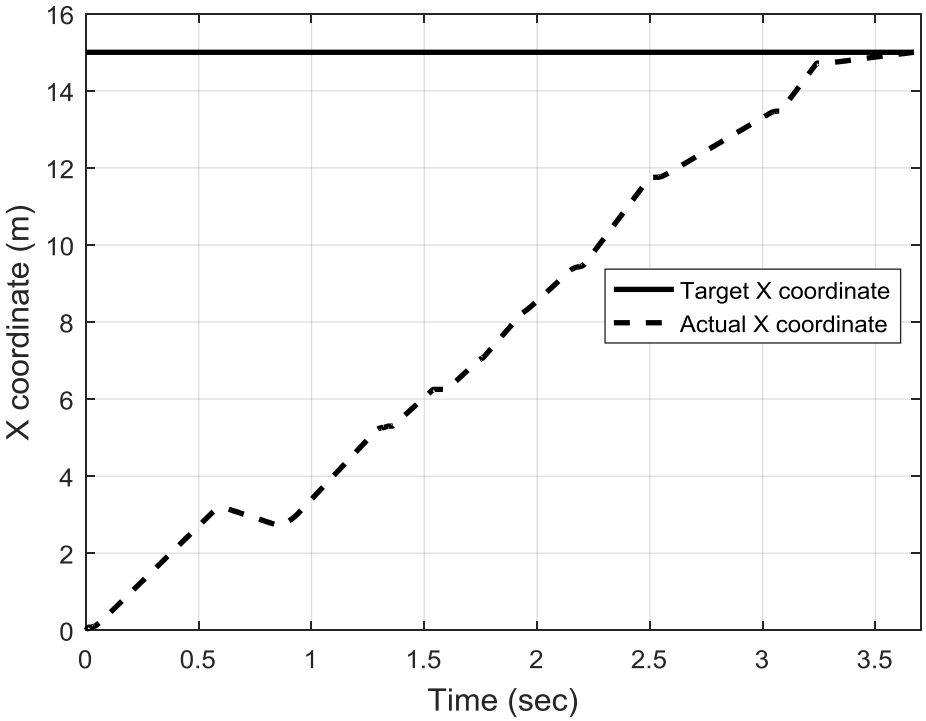



Fig. 7.22 X-coordinate of UGV whilst navigation in case study-II using ANFIS.

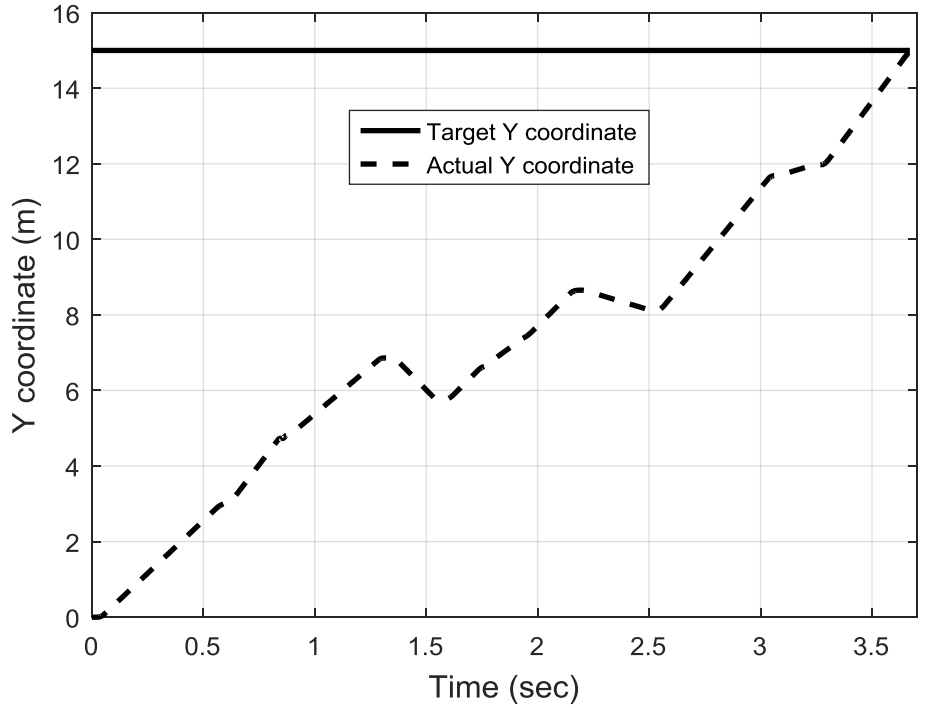



Fig. 7.23 Y-coordinate of the UGV whilst navigation in case study-II using ANFIS.

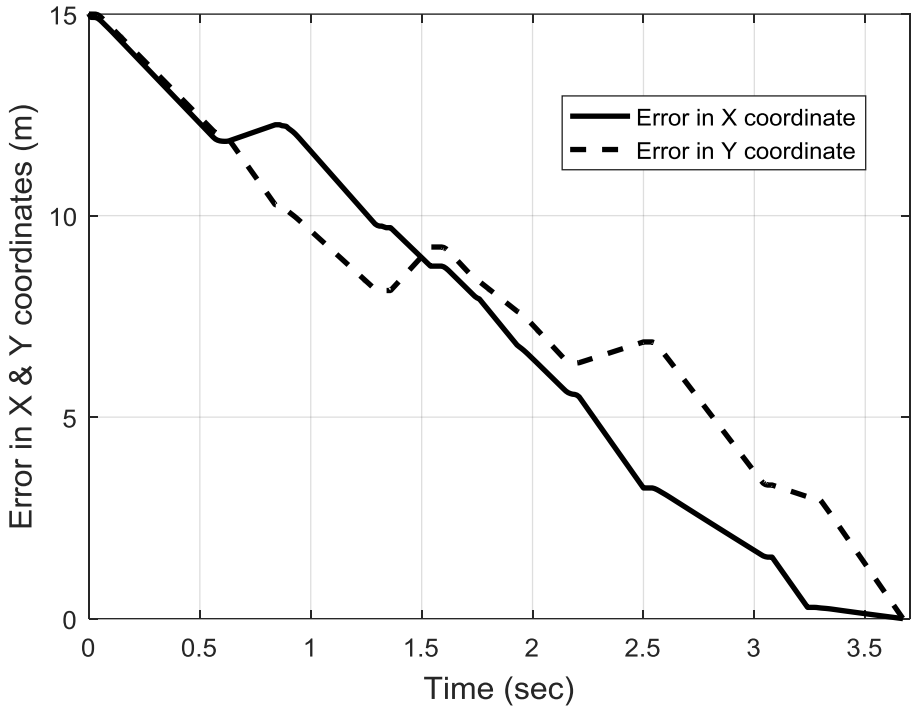



Fig. 7.24 Tracking error in X and Y coordinates of UGV in case study-II using ANFIS.

## 7.6 Comparison with the related work

To prove that our proposed ANFIS model has improved the path planning and obstacle avoidance signifcantly, a comparsion is presented with a previous study that utilises a similar ANFIS model in the state of the art as demonstated in Fig. 7.25. In the conducted ANFIS model based on the literature, the motion of UGV has not been able to establish a feasible path that connects between the initial and target points. This indicates a need to develop an ANFIS model that can achieve a smooth navigation without undesirable uncertainties in the operational performance. The comparison has clearly shown that our proposed ANFIS controllers have improved the navigation response significantly in terms of obtaining an optimal path and reducing the elapsed navigation time.

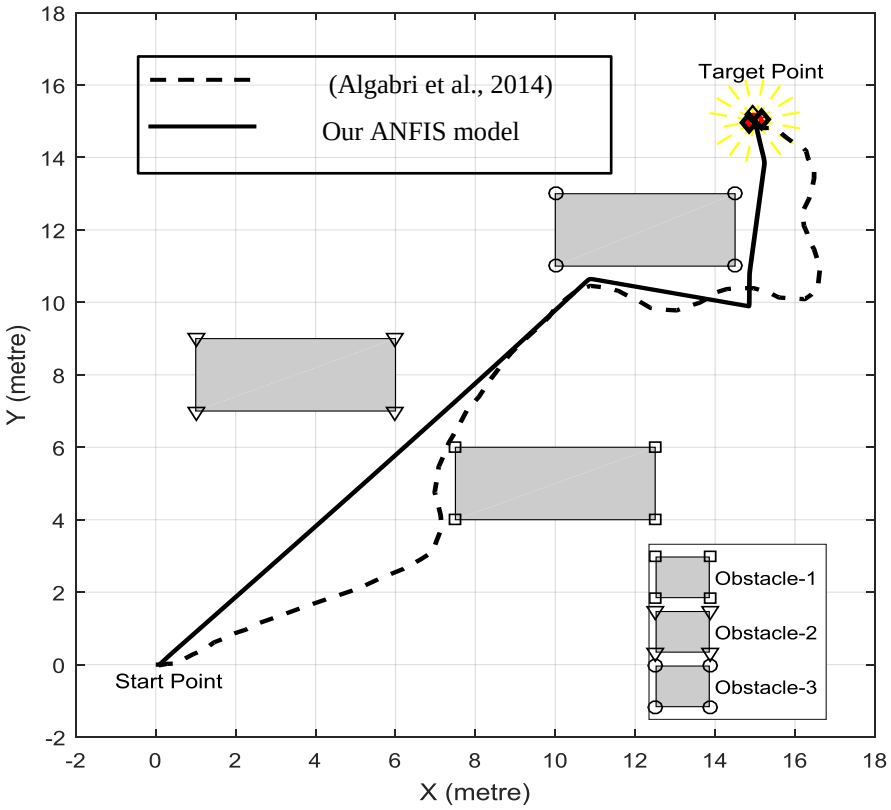



Fig. 7.25 Comparison navigation platform contains three static obstacles with similar sizes.

## 7.7 Chapter Summary

In this chapter, an adaptive neuro-fuzzy inference system has been implemented to control the angular velocities of the wheels of an unmanned ground vehicle. These velocities guide the vehicle successfully and safely to reach the destination in an unstructured environment without colliding with the obstacles presented on its path. The design system comprises four ANFIS controllers. Two of which are used for the target reaching to ensure that the UGV reaches its destination coordinates. The other two are utlised in guiding the UGV to avoid the collision with obstructing obstacles. The four controllers are integrated to provide the right and left angular velocities. The validation of the proposed method has been demonstrated by considering two case studies; firstly, seven identical static obstacles are placed randomly with the workspace; secondly, eleven obstacles have considered with different sizes and shapes. In case of reducing obstacle numbers significantly, the obstacle avoidance ANFIS controller did not activate until the vehicle confronted an obstacle. Therefore, the target reaching ANFIS controller is activated most of the time. This case is similar to a situation when the path is clear of obstacles so that the UGV moves straightly between both start and target points.

# Chapter 8

# Experimental Work Based on Real Time Navigation of UGV

## 8.1 Introduction

In this chapter, an unmanned ground vehicle is implemented practically to perform real-time navigation. The implementation has been accomplished by enabling the UGV of interacting and planning its motion in an unknown environment. In order to achieve a full autonomous architecture of the UGV, the latter has to be equipped with sensors, which are capable of sensing external surroundings and internal status of the UGV. The implementation process of the UGV's architecture will be discussed in details in the following sections, which demonstrate the components of each stage and their function. Additionally, several algorithms are introduced to create a communication link between any device and another. Furthermore, the fuzzy inference system controllers given in Chapter 6, are utilised to perform the functions of obstacle avoidance and target reaching. The obstacle avoidance controller is constructed based on sensors' information. Thus, it is needed to provide distance information to calculate how far objects are from a UGV's platform. For the target reaching controller, the orientation is needed to know at any time of the UGV's movement in a workspace. To reach any destination, a compass is embedded in the UGV's platform to determine the heading of navigation instantaneously. The FIS is rewritten using Python programming language based on real time experiments to investigate its performance.

## 8.2 Chapter Organisation

This chapter is organised as follows: In the following section, a brief overview of the UGV's architecture is presented based on our design. Section 8.4 is dedicated to the sensor technology utilised in the architecture. The microcontrollers and the communication protocol between Raspberry Pi and Arduino are described in Section 8.5. In Section 8.6, the motion control is explained based on the driving actuators. The control methodology is discussed in Section 8.7. The experimental results are introduced in Section 8.8. Finally, the chapter summary is given in Section 8.9.

## 8.3 Architecture of Unmanned Ground Vehicle

To construct a UGV's architecture that operates in a workspace without an on-board human presence, on-board sensors are required to be equipped onto the platform to make it capable of moving autonomously. Based on an operational observation, the UGV makes full decisions about its behaviour in response to its surroundings. The architecture of the UGV is constructed mainly based on four functions that must be provided to enable an autonomous navigation. Firstly, the perception is to enable the UGV of understanding the surroundings, its current orientation and position within the workspace. The perception is achieved based on sensor technology. Three essential sensors are provided to make the perception occurring based upon the distance detection of surrounding objects. A compass module is equipped to recognise the intended direction of the UGV. A quadrature encoder is embedded into the motors' shafts. It is necessary to specify the direction and the speed of the wheels. Accordingly, travelled distances and elapsed times can be obtained. They are the key elements for localising the UGV instantaneously within any given environment.

Secondly, master and slave microcontrollers communicate with each other based on values of sensor information to make appropriate actions accordingly. The Arduino, the slave microcontroller, is directly integrated with the three distance sensors. It constantly sends the sensory information to Raspberry Pi via coded strings. Conversely, the Raspberry Pi also continuously sends commands back based on particular aims according to required tasks. Based on the communication between Raspberry Pi and Arduino, the latter will send signals to the driving circuit. This is the third stage of the architecture and it provides the required power for the actuators. Fourthly, the actuators drive the UGV forward/backward, turning left/right, or stop based on a particular scenario in a given workspace environment. The proposed architecture of the UGV is demonstrated in Fig. 8.1.

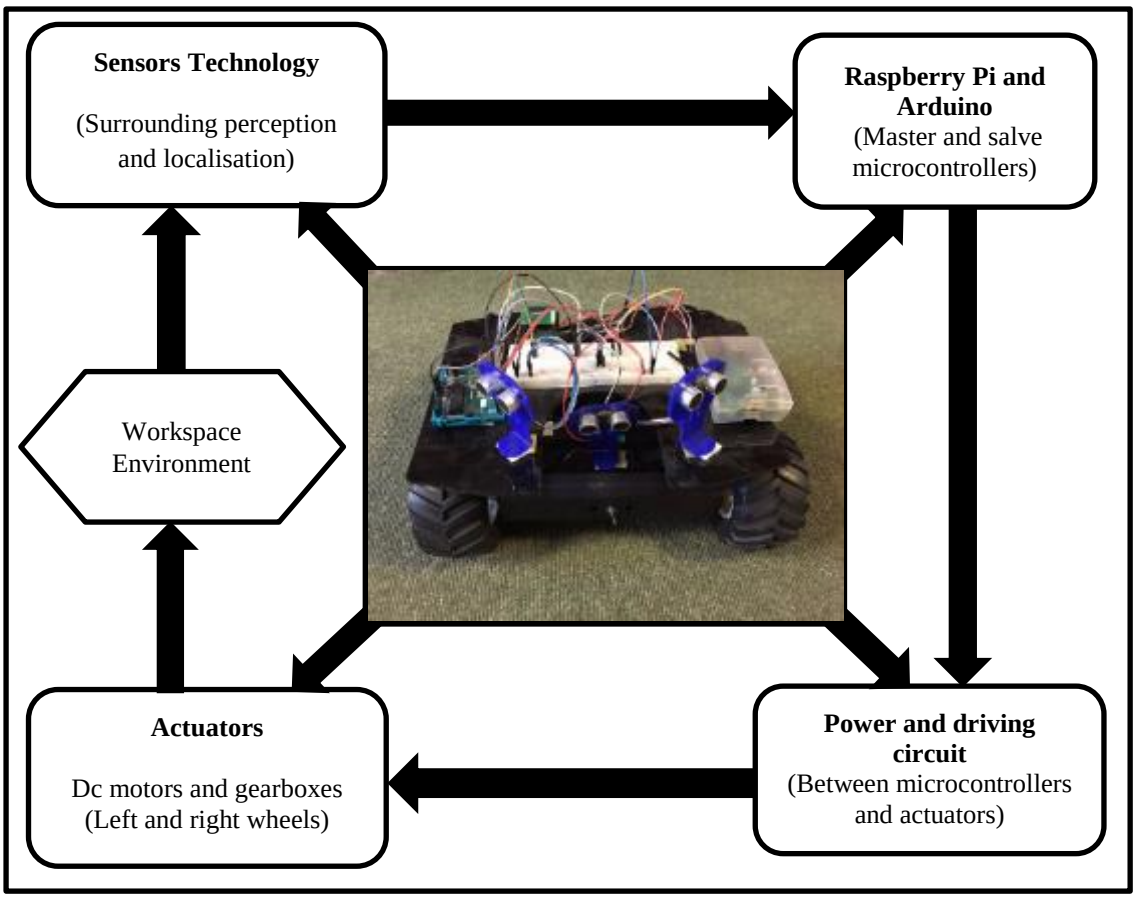



Fig. 8.1 Architecture of unmanned ground vehicle.

## 8.4 Sensor Technology

Sensing technology is the key element in understanding and sensing external environments and an internal state of unmanned ground vehicles. The sensing of surrounding can be achieved by using various sensors. One of the most important tasks of an autonomous platform is to acquire knowledge about its environment. The sensors take measurements and translate measured information to meaningful data. When sensors collect information from the real world environment, they are called exteroceptive sensors based on perform functions. For instance, for obstacle avoidance, sensors utilise for detecting distance with respect to surrounding obstacles. There are also some sensors called proprioceptive sensors, which they are used to measure the internal values of a system such as wheels' speed and battery status. In this chapter, three types of sensors are used to guide an unmanned ground vehicle in each workspace. These sensors are explained in detail in the following sections.

### *8.4.1 Ultrasonic Range Sensing*

An ultrasonic sensor has been the most popular sensor modality for autonomous systems because not only it can be readily interfaced with microprocessor systems. However, it is also probably the least expensive approach for "real time" sensing. In addition, the ultrasonic sensor requires a minimal power. Additionally, it offers precise ranging information from roughly 3 cm to 4 metres, thus, that makes it an ideal range for robotic applications. The characteristics of ultrasonic range sensors are not unlike those associated with other range sensors, which use different parts of the electromagnetic spectrum. Such sensors are optically based or millimetre wave range or radar sensors from specular reflections, absorption, bandwidth, resolution, etc. that will affect the final measurement. Three ultrasonic sensors, type SR04, are used for distance detection of surrounding obstacles. The sensors are placed in three positions within the UGV's platform i.e. left, front and right positions. Fig. 8.2 demonstrates the sending and receiving of trigger and echo signals. It is observable that the transducer creates an ultrasonic sound, which travels from the sensor, and then it bounces back to an object's surface. The theory of how such sensors can calculate the distances based on the trigger and echo signal is illustrated in Algorithm 8.1. It transmits a pulse of sound outside the range of human hearing at 40 KHz. This pulse travels at the speed of sound away from the ranger in a cone shape and the sound reflects back to the ranger from any object in the path of sonic wave. The ranger pauses for a brief interval after the sound is transmitted and then awaits the reflected sound in the form of an echo. When the trigger signal is sent and the echo signal is received, the distances to objects can be computed based on elapsed time. The ultrasonic sensor is integrated with the Arduino microcontroller using four pins. Two of which pins are utilised for power supply. The other two pins are for the trigger and echo signals (Nedelkovski, 2015). The scheme diagram of connection between the ultrasonic sensor and Arduino is shown in Fig. 8.3.

**Algorithm 8.1:** Distances calculation between the UGV and an object

**Inputs:** Trigger the ultrasonic sensor to transmit a signal, receive an echo signal, the speed of the sound is known in the air, 340m/s.

**Outputs:** Calculate the distance to an object.

1 **if** an object within the sensing range **then**
2 Calculate the time (T) difference between sending and receiving the sound pulse.
3 (The ultrasonic sensor measures the distance by timing how long for an ultrasonic wave sent out by an emitter to bounce off an object and come back to the receiver.)
4 Distance = (T x Speed of Sound) / 2
(The '2' is in the formula because the sound has to travel back and forth).
5 **else**
6 No obstacle is existed within the sensing range.
7 **end**

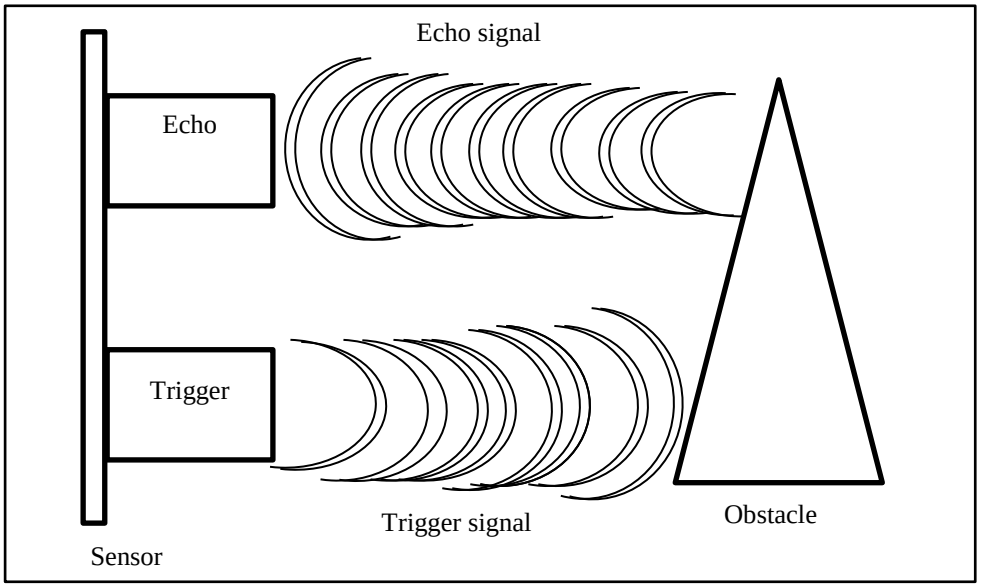



Fig. 8.2 Ultrasonic sensor transducers sending and receiving signals.

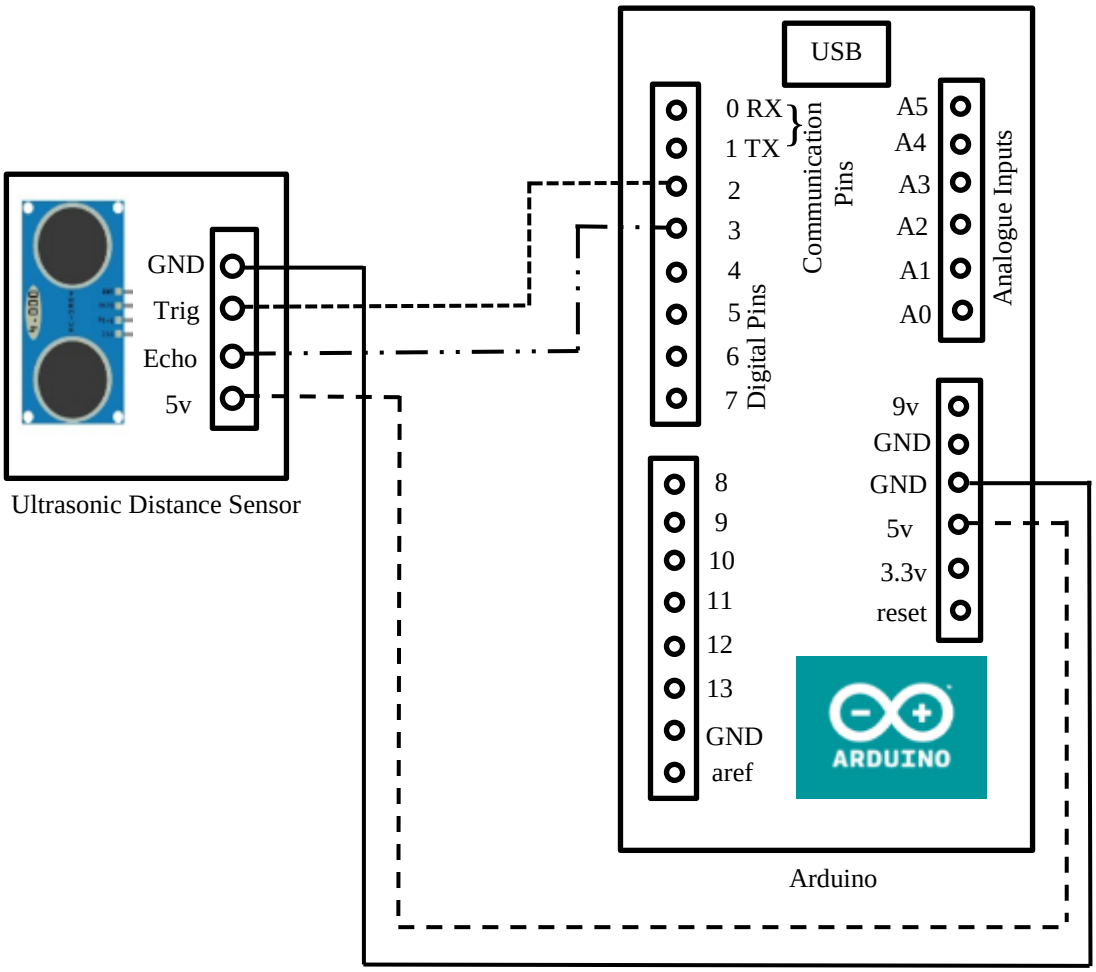



Fig. 8.3 Interfacing between ultrasonic sensor and Arduino.

### *8.4.2 Magnetic Compass Module*

In this setup, a magnetic compass module is used type CMPS03. It has been specifically designed for aiding the navigation. The heading control can be achieved by producing a unique number that represents the orientation of a robotic platform. The connection of the CMPS03 compass module and Arduino is demonstrated in Fig. 8.4. There are nine pins in the configuration of the compass. Nonetheless, only four pins are used to be integrated with the Arduino. Pins 1 and 2 energise the compass by 5V power supply. Pins 2 and 3 of the compass are utilised for a communicative purpose. The compass uses a magnetic field sensor type Philips KMZ51, this sensor is sensitive enough to detect the Earth's magnetic field. It is used to compute the direction of the horizontal component of the earth's magnetic field.

An inter-integrated circuit (I2C) is a serial protocol that is used for communication between the compass module and Arduino microcontroller. This is based on the serial clock (SCL) and serial data (SDA). The SCL is the clock signal which synchronises data transfer

between devices on the I2C bus. The other line is SDA which carries the data. The two lines are open drain, which means that pull-up resistors are needed to be attached to these lines in order to make the line high, because the devices on the I2C bus are active low. The commonly used range of the pull-up resistors are varied from 1.8-2 KΩ. The data signal is transferred in sequences of eight bits. The first eight-bit sequence indicates the address of the slave to which the data is sent.

The Arduino communicates with the magnetic compass by issuing a start sequence on the I2C bus. A start sequence is one of two special sequences defined for the I2C bus, the other being the stop sequence. The start sequence and stop sequence are special in that these are the only places where the SDA (data line) is allowed to change whilst the SCL (clock line) is high. When data is being transferred, the SDA must remain stable and not change whilst SCL is high. The start and stop sequences mark the beginning and end of a transaction between the Arduino and the magnetic compass module. Data is transferred in sequences of 8 bits. The bits are placed on the SDA line starting with the most significant bit (MSB). The SCL line is then pulsed high, then low. Algorithm 8.2 illustrates the structure of how the magnetic compass will be determining the orientation of the navigation.

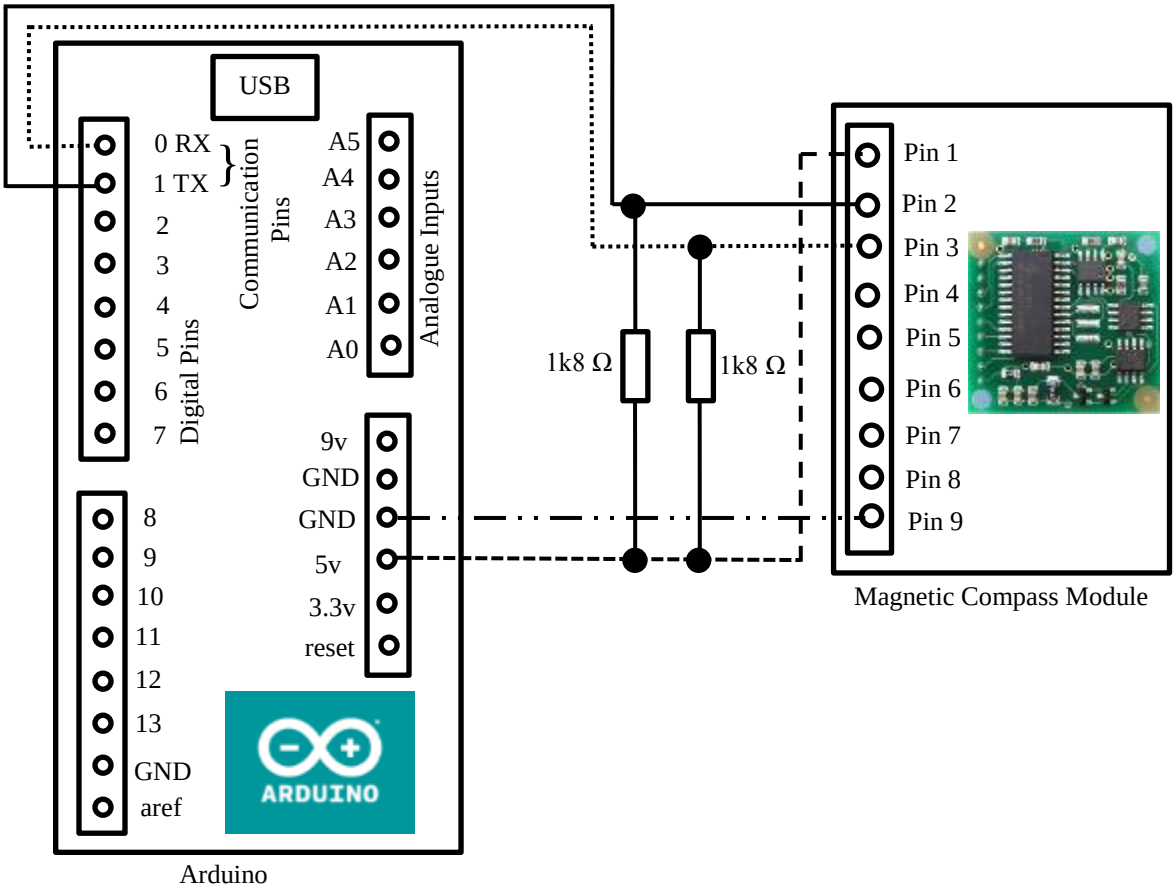



Fig. 8.4 Interfacing between compass sensor and Arduino.

**Algorithm 8.2:** Determining the orientation of the UGV

**Inputs:** Define the address of the compass and the baud rate (bit rate per second).

**Output:** The orientation of the UGV.

```
1 Loop
2 |  Communicate CMPS03 and Arduino using I2C protocol
3 |  Send registers
4 |  While there is a byte to receive
5 |  |  Read the hightByte
6 |  |  Read the lowByte
7 |  |  The orientation = ((hightByte<<8) + lowbyte)/10
8 |  end
9 end
```

### *8.4.3 Quadrature Encoder*

A quadrature encoder is classified as a proprioceptive sensor that can measure internal values of the UGV i.e. the speed and direction of a rotating shaft. It is also known as an incremental rotary encoder based its operational performance. The Quadrature encoder can utilise an optic sensor or other type to perform its function. Regardless the type of utilised sensors. The quadrature encoder produces two outputs of square waveforms that are shifted by 90 degrees from each other. The amplitude of the square waveforms varies from 0-5 V. The speed of the rotation can be measured by using only one output of the quadrature encoder. However, in order to determine the direction of the rotation, the two outputs will be used. This can be achieved based on a pattern of binary numbers generated by both outputs. The quadrature encoder is essential for odometry which it is a method used for collecting data to estimate the change in the position of the UGV after it has moved over time. The connection between the compass module and Arduino is demonstrated in Fig. 8.5 below. The presented algorithm for determining the travelled distance is illustrated in Algorithm 8.3.

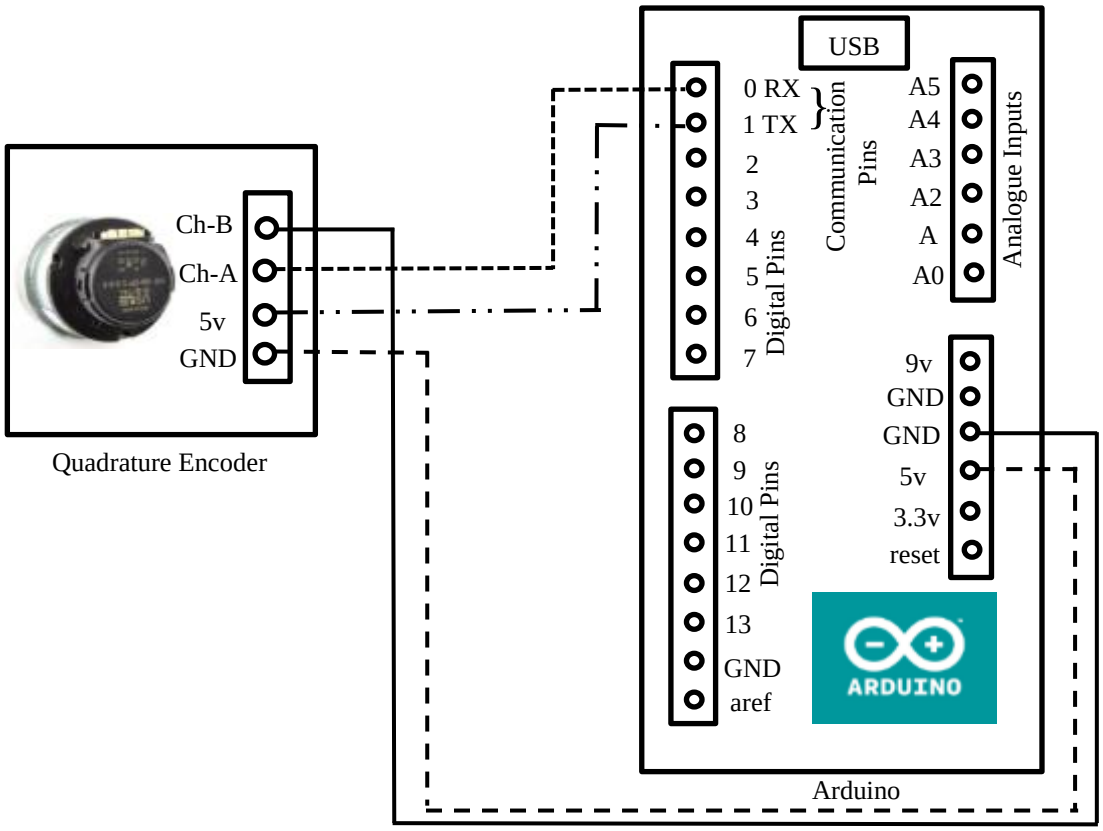



Fig. 8.5 Interfacing between quadrature encoder and Arduino.

The optic disk inside a quadrature encoder contains two tracks denoted channel A and channel B. These channels are shifted by ninety electrical degrees out of phase from each other as illustrated in Fig. 8.6. This is the key element that provides the quadrature encoder its functionality. For the direction sensing, a controller can determine the direction of movement based on the phase relationship between channels A and B. When the quadrature encoder rotates in a clockwise direction, its signal shows channel A leading channel B. Reciprocally, the reverse occurs when the quadrature encoder rotates anticlockwise.

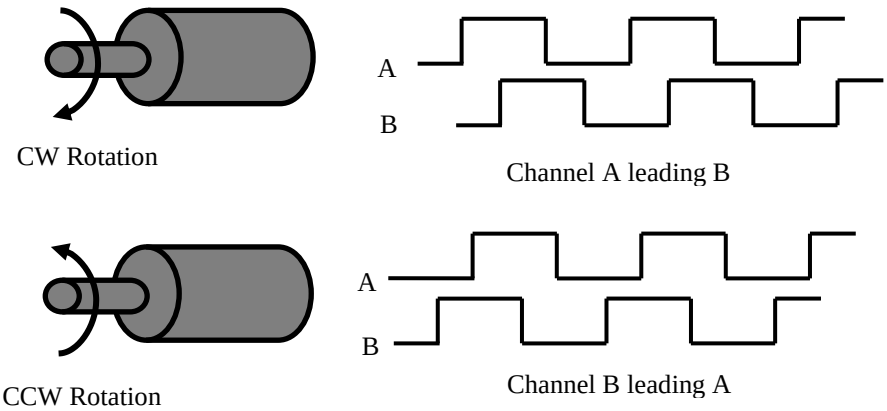



Fig. 8.6 Pulses of channels A and B in CW and CCW movement.

**Algorithm 8.3:** Travelled distance calculation

**Inputs:** Radius of a wheel (r), revolutions per minute of the wheel based on the quadrature encoder measurements.

**Outputs:** The travelled distance of the UGV.

1 **if** revolutions per minute is measured by the encoder **then**

2 Calculate the circumference of the wheel from its radius (Circumference=2rπ).

3 Calculate the speed, speed = revolutions per minute x circumference

For *k* rpm, speed = $k\frac{rev}{min}x\frac{1min}{60\,s}x\frac{2\pi r}{1\,rev} = k\,\frac{m}{s}$

4 Calculate the timing interval, the time is calculated using a built in function in Arduino programming platform, the function is called millis()

5 Calculate the travelled distance, travelled distance = speed x time

6 **else**

7 No movement is occurred.

8 **end**

The paramount purpose of using quadrature encoder is for localisation of the UGV. If the initial position of the UGV is assumed known, the next position can be determined by using the quadrature encoder. This process is so called the dead reckoning, in which the new position of the UGV can be predicted and updated based on the continuous movement of the UGV. A systematic design of the UG can dramatically eliminate the deterministic errors of localisation. It is required to adjust the alignment of the wheels and diameters of wheels are equal. In addition, the floor contact of a workspace should be smooth and flat to avoid slippage as a result of variation in the contact point of the wheels.

## 8.5 Microcontrollers and Communication Protocol

Microcontrollers represent the core elements of the robotic platform. They perform the function of processing and analysing data. They comprise of multi general purpose input/output (GPIO) pins that can be integrated with sensors or peripheral devices. In this work, two microcontrollers are used i.e. Raspberry Pi and Arduino based on the current advances in electronic technology. The Rasberry Pi and Arduino have been increasingly used in many applications and they have become quite popular. Recently, these controllers are increasingly utilised instead of traditional microcontrollers. In Raspbey Pi, the operating system is based on the Linux system and the distribution is based on the Debian operating system. Based on Linux

systems, many programming software can be used to perform a coding task for a specific problem. In particular, Python software which is already built-in and it is included in the distribution package. Nonetheless, other software can be installed and used based on a user's desire.

Similarly, Arduino is another platform used in many applications of embedded systems. It is an open-source electronic prototyping platform that enabling users to create interactive electronic objects. The programming based on Arduino platform can be accomplished using C/C++ software packages. Raspberry Pi is utilised as a master microcontroller and Arduino is used as a slave microcontroller. The master controller. Such multi-stage controllers enable the embedded systems of sharing tasks and programming threads to minimise the computational time and provide more ports and units for transferring and processing data.

There are various ways to connect Raspberry Pi and Arduino, such as using GPIO pins, serial peripheral interface (SPI), I2C bus and universal serial bus (USB). However, the USB connection can be considered as the easiest way to achieve the interfacing, because required hardware is minimal. The USB is a serial protocol for communication between the Raspberry Pi and Arduino (Oscar, 2013), which is the simplest communication approach. The interlinking between both microcontrollers is demonstrated Fig. 8.7.

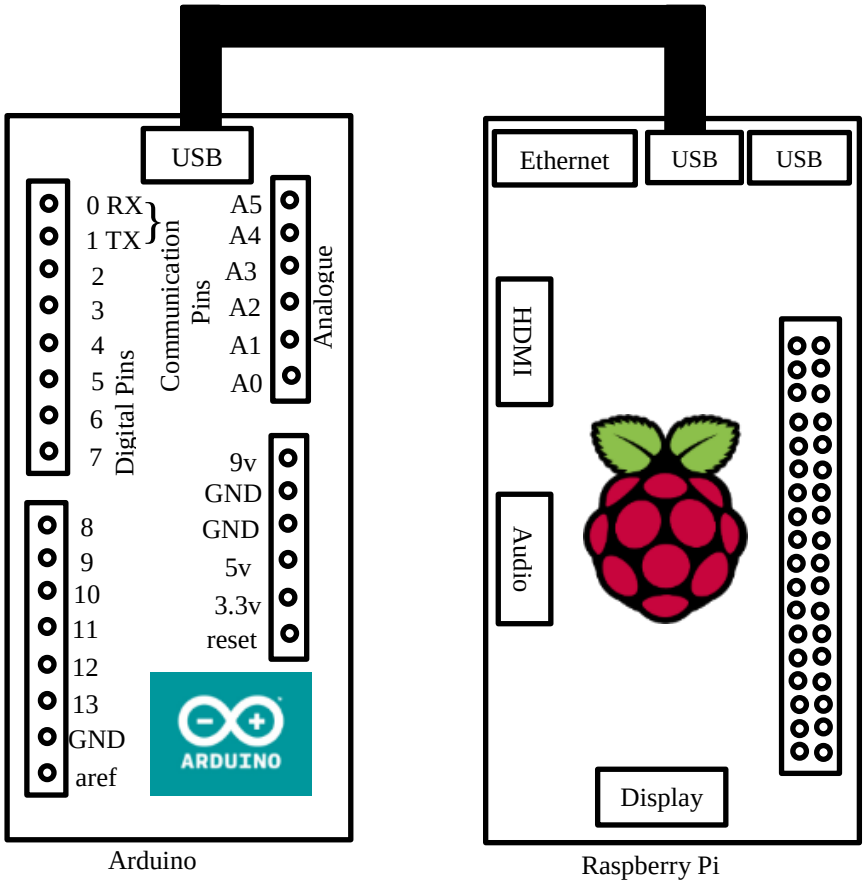



Fig. 8.7 Interfacing of Raspberry Pi and Arduino using USB connection.

In order to make the Raspberry Pi and Arduino taking to each other, a communication protocol is needed. It involves sending and receiving data between Raspberry Pi and Arduino. The communication protocol is based on teletypewriter (TTY). The TTY allows alphanumeric character to be typed in and sent synchronously. Therefore, an algorithm is proposed for the communication between Raspberry Pi and Arduino. The principle of the algorithm is based on encoding and decoding characters of strings. For instance, if a string of numbers and letters is coded and sent visa a single combination of numbers and letters, the string will be received on the other end in order to be decoded to make use of its data. The data sending can be from Raspberry Pi to Arduino or vice versa.

On Raspberry Pi side, a string is generated that comprises three parameters i.e. "M, moving, turning" to be sent via a single command to Arduino. This provides an order to Arduino to make an action for motor to either moving forward or turning left/right. On Arduino side, a single string is also generated for distance reading of three sensors to be sent to Raspberry Pi. This makes and closed loop circuit of communication. When the Raspberry Pi receives the sensors' reading, it analyses the string to recognise which sensor requires more considering based on its provided distance. The master controller i.e. Raspberry Pi, will be constantly sending and receiving data. Likewise, Arduino is the slave microcontroller, will set the motors' motion according to the specified commands specified by the sent string. The description of the communication process is described in Fig. 8.8 flowchart.

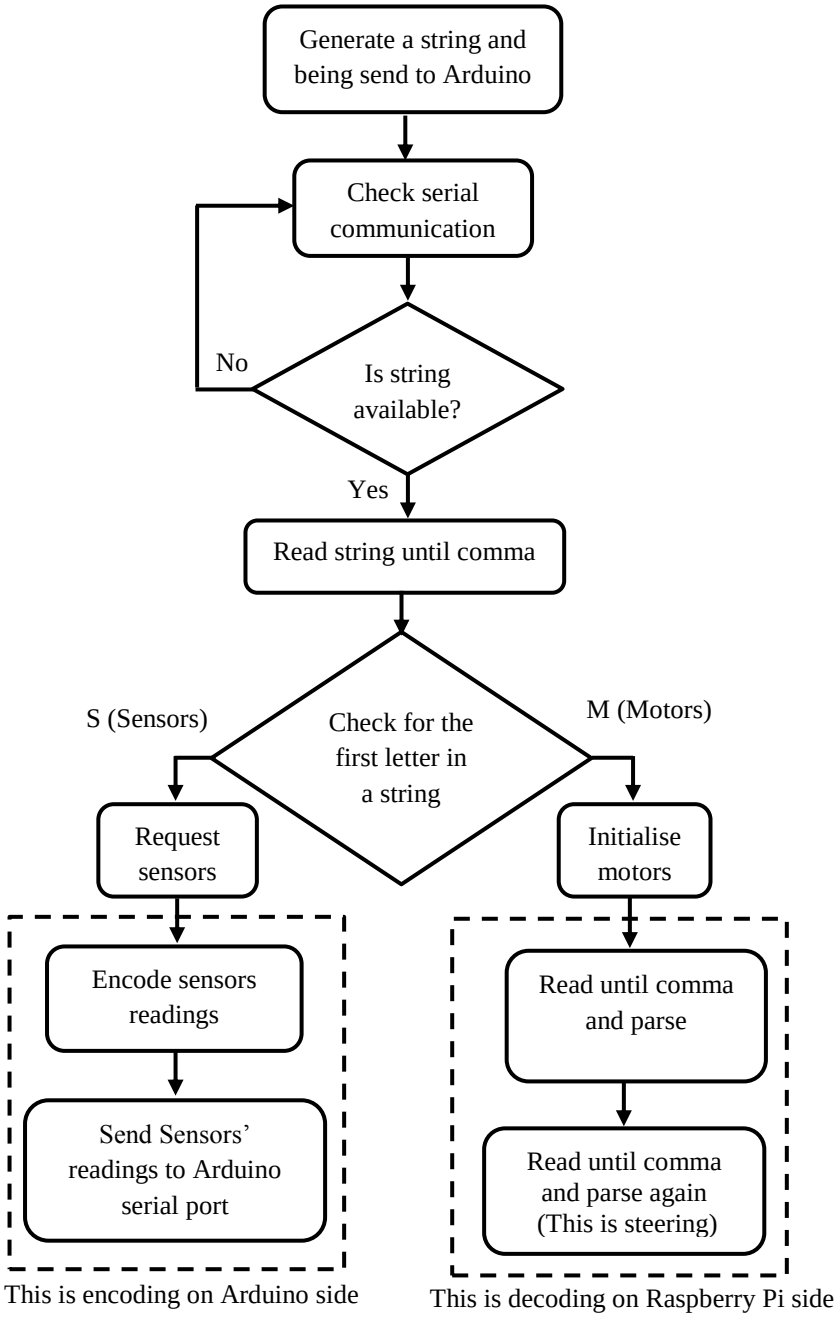



Fig. 8.8 The communication protocol algorithm.

## 8.6 Motion Control

The automatic controlling of the wheels' movement is essential for an autonomous naviagtion. A driving power circuit is required to be an intermediate connection between the actuators and the Arduino microcontroller. The driver circuit is designed based on a synchronous regenerative motor drive type Sabertooth 2x10 (dual motor driver of 10A output current each). It has two main channels; one is used for forward/backward motion. The other one is used for controlling the steering. The tolerance voltage of this motor driver circuit is from 6V to 25V. The output current is up to 10A per motor. The schematic diagram for the

driver motor circuit and the wheels is demonstrated in Fig. 8.9. The battery provides a 12V DC power supply. It supplies the motor driver circuit with the required power supply. The regenerative motor drive provides the motors on the left and right side its rated voltage individually, i.e. 12 V.

In addition, the driving circuit receives two control signals from the slave microcontroller "Arduino". The control signals are used either for moving forward/backward or for steering the UGV to control its orientation. This can be achieved by programming the Arduino device internally to supply an operation signal in order to provide a certain type of movement. The deterministic motion of the wheels on both sides are driven differentially, which means the movement is based on two separately driven sides. Hence, the direction of the UGV can be controlled by varying the rotation rate of the wheels a specific side. If the wheels move at the same speeds the UGV will traverse straight forward. However, if the wheels on the left side have less rate of rotation, it leads of turning the UGV in the right direction. Similarly, when the wheels on the right side have less rate of rotation, the UGV will turn left.

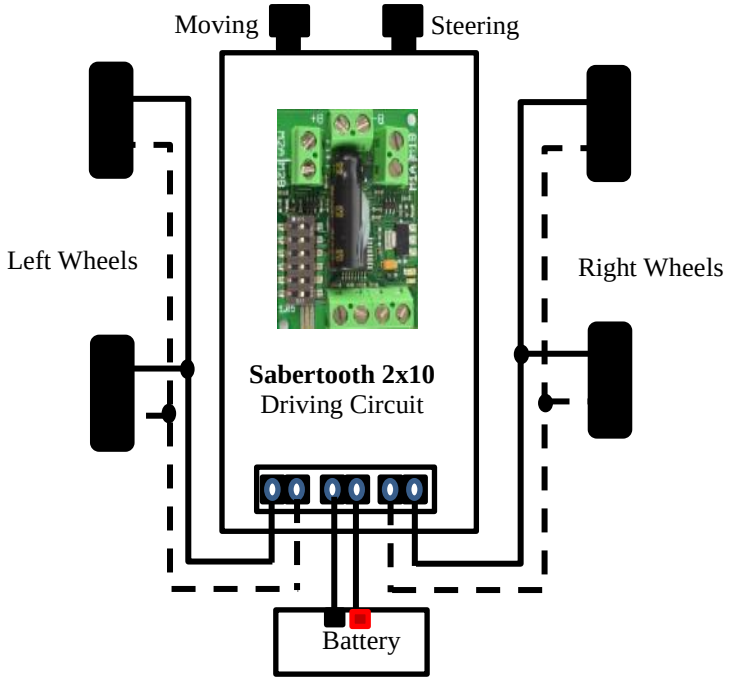



Fig. 8.9 The schematic diagram for the driving motor circuit and the wheels.

The actuators of the UGV can be driven based on the width of supply signals. In principle, the actuators are combined of the gearboxes, DC motors driving circuit and encoders. The DC motors in this platform are interfaced with the motor driving board (Sabertooth 2x10) that acts like two servo motors in an independent mode of servo control. Therefore, you need to send

servo type signals to control the motor controller, which in turns converts those to regular DC motor signals with higher currents/voltages.

The servos are controlled from full speed reverse (1000 us) to idle (1500) to full speed forward (2000). Whereas, the steering is controlled from full turn (left or right) at 1000 us, centre (no turning) at 1500 us and full turn (other direction, depends on the wiring of motors) at 2000 us. In case of forward and backwards movements, all wheels are turning in the same direction simultaneously. The pulse length (in microseconds) determines the position of a servo motor. In the case of the Sabertooth motor controller, instead of determining the position of the motor, the value is used to determine the throttle on one channel and the turning on the other channel. The waveforms of digital signals for servomotors are shown in Fig. 8.10.

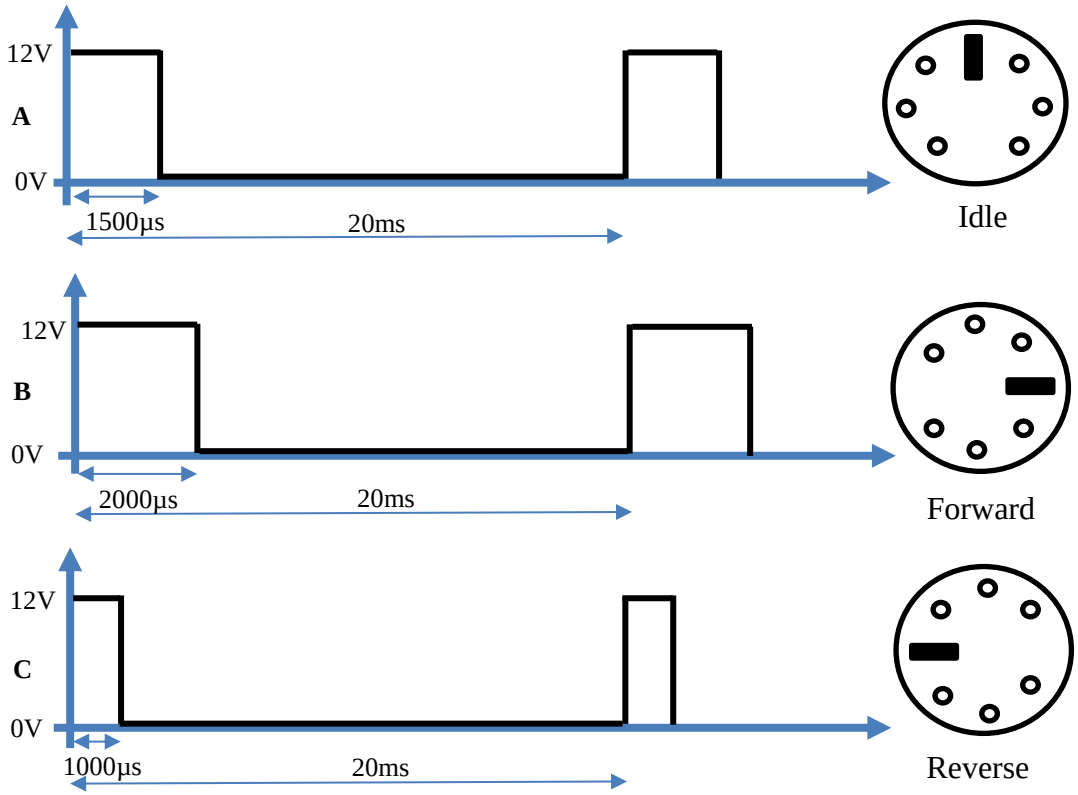



Fig. 8.10 Servo control signals.

The pulse width modulation (PWM) is used to control the width of pulses to a servomotor to change its rotation angle. It can be also utilised a variety of applications including sophisticated control circuitry. The modulation of signal's width can be achieved based on a duty cycle. The duty cycle can be modified based on a percentile ratio whilst the signal becomes high or low over a period of time. This period is the inverse of the frequency of a waveform. The digital signals for the specified period of time will be either ON or OFF correspondingly.

The generated digital signals from Arduino microcontroller is conducted by using the PWM to change the frequency to 50Hz (or period of 20ms). Hence, the duty cycle is changed to 7.5% (or length of the pulse of 1.5 ms = 1500 us). Between 5% duty cycle and 10% duty cycle at 50Hz, a full range of 1.0-2.0 ms (or 1000-2000 us) can be achieved.

## 8.7 Control Methodology

The applied control methodology in the experimental work is based on the fuzzy inference system controllers introduced previously in Chapter 6. The controllers are coded in the master controller, Raspberry Pi, using the Python software package. For the obstacle avoidance FIS, the universe variables are generated for both inputs and outputs. The inputs are the three distance sensors and the outputs are the driving motors of the left and right sides. Two trapezoidal-shaped membership functions are for each input, the membership functions are used to cover the range of the nearest distances, (0-50cm), and far distances, (50cm-1m). In addition, five singleton membership functions are used for each output to cover the specified intervals of the driving motors of the left and right wheels. The moving back quickly is achieved by applying 2000 μs signal width. Also, for moving back but slowly is fulfilled by reducing the width of the applied signal to 1750 μs. The idle case of the obtained when the width of the applied signal equals 1500 μs. Similarly, for the moving forward quickly and slowly can be accomplished when the applied signals are 1000 μs and 1250 μs, respectively. Consequently, fuzzy rules based on an inference engine are created to mimic the behaviour of the UGV operation according to sensor data and thus, the driving motors will respond accordingly to control the motion of the wheels. Sixteen fuzzy rules are created in the obstacle avoidance FIS.

The target reaching is implemented using the same principle. However, another input is added to the target reaching FIS i.e. target distance in additional to the target angle. The target distance is to provide an indication when the target is reached. Thus, the driving wheels will be stopped. The target angle covers a wide range of angles in the clockwise and anticlockwise from (-180 to 180 degrees). It has been arranged into five triangular-shaped membership functions. The two outputs of the target reaching FIS is also fuzzified into five singleton membership functions to make an action corresponds to a specific case. Like to the obstacle avoidance FIS, the fuzzy rules are generated to make proper decisions based on the inference engine. The number of generated fuzzy rules are twenty in total that would cover a high range of the possible scenarios. As explained earlier, the Rapsberry Pi is the master controller, which

will send commands to Arduino based on the sensing data, the Arduino in turn, will provide the signals to the driving unit of the designed architecture.

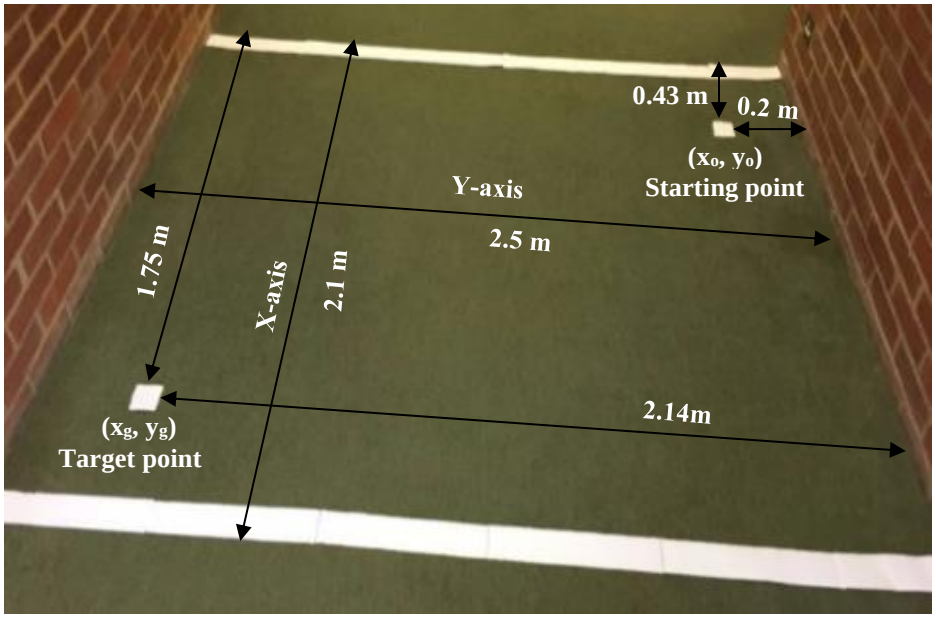
## 8.8 Experimental Results

The experimental results are provided to validate the applicability of the developed techniques and algorithms. The solvability of the obstacle avoidance and target reaching problems is considered practically based of fuzzy inference systems presented in Chapter 6. The experiment results are conducted based on into three case studies. The UGV has to pass obstacles if any and reach the destination successfully and feasibly in each case study to prove the effectiveness of the designed architecture and proposed algorithms.

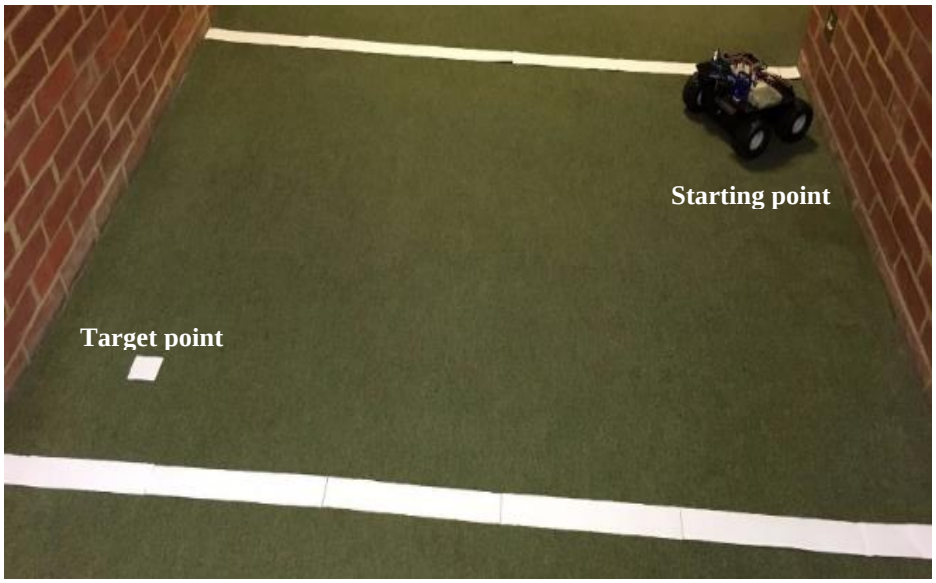
### *8.8.1 Case Study-I*

In this case, there are no obstructing obstacles considered to intercept the UGV's movement. The main purposes of such a study are to ensure that the UGV can move toward its orientation and stop when it reaches the destination; to determine the travelled distance and the localisation of its posture instantaneously. Consequently, it can localise the coordinates of the destination. It has been assumed that an initial position of the UGV is already known and the target point is specified. Observably, the UGV has followed a straight line that connects the start and the target points. The length of the travelled distance is 2.2 m at an elapsed time equals 1.95 seconds. The speed of the UGV equals 1.128 m/s. The difference between the actual coordinates and the target point determine instantaneously whether the UGV has reached its destination or not. The direction of the UGV is placed directly towards the destination. Hence, it is not expected that the UGV makes any changing in its heading whilst moving because no obstructing obstacles are existed to confront the movement.

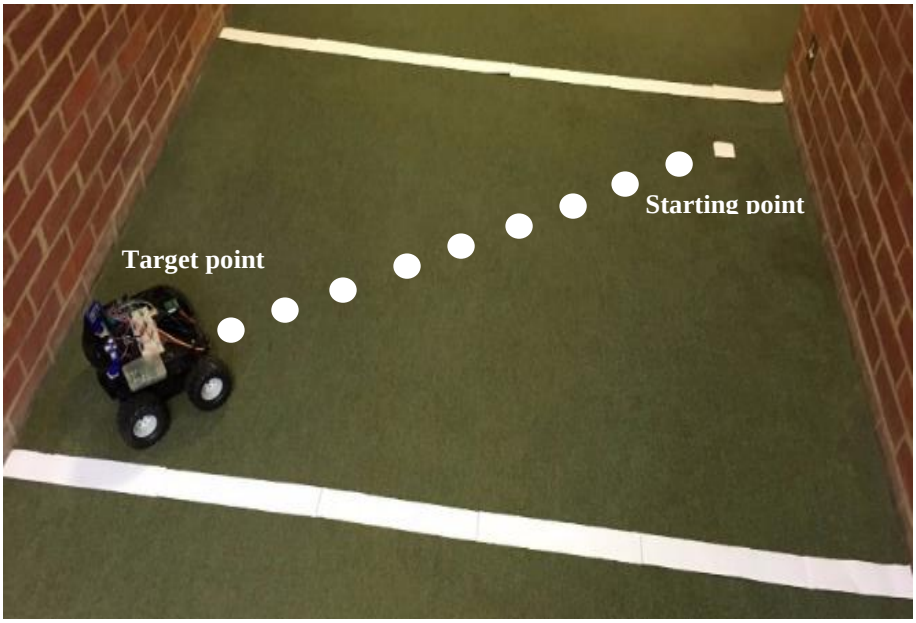
The revolutions per minute of the wheels determine incrementally the travelled distance. Hence, when the coordinates of the starting and the target points are equal, the stopping instance occurs. Fig. 8.11 demonstrates this typical case study into three parts; (a) the allocated workspace which is dimensioned at 2.5 by 2.1 metres in addition to the coordinates of the starting and target points, i.e. (0.43, 0.2) and (1.75, 2.14), respectively, (b) when the UGV is positioned at the starting point and (c) when the UGV reaches the target point. In fact, such a case study can be considered the simplest scenario where the UGV might operate on.


0.43 m
0.2 m
(xo, yo)
Starting point
Y-axis
2.5 m
1.75 m
X-axis
2.1 m
2.14m
(xg, yg)
Target point

(a)


Starting point
Target point

(b)



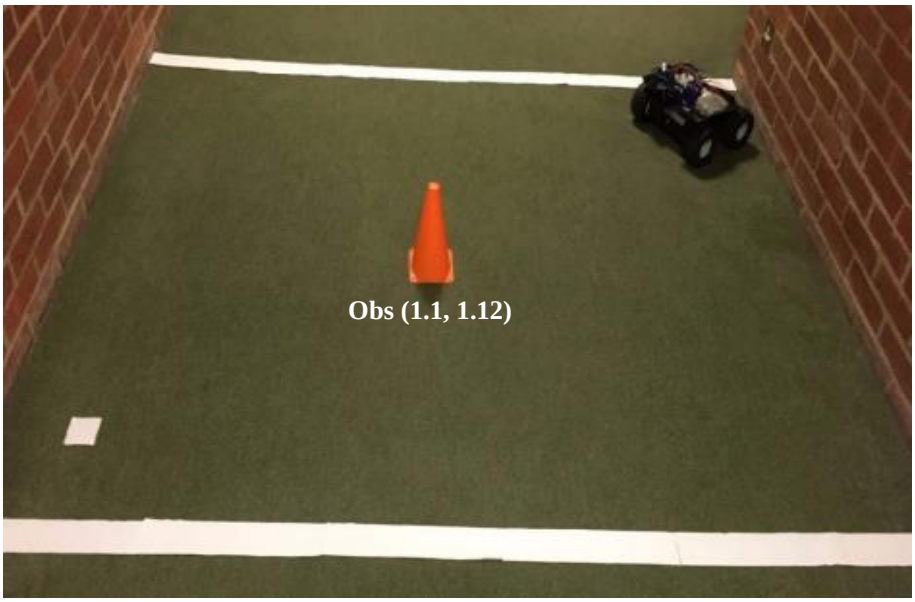

(c)

Fig. 8.11 Case study-I when no obstacles are existed; (a) the mapped workspace, (b) the UGV at the start point (c) the UGV at the target point.

### *8.8.2 Case Study-II*

In this case study, a scenario is considered when only obstacle is placed at a random distance between the starting and the target points. By this case study, the complexity is gradually increased to guarantee that the UGV is capable of performing the obstacle avoidance and target reaching as required based on a particular situation. The starting and target points are considered as in the previous case study. The UGV commences its motion by headings towards the target as shown in Fig. 8.12(a). At this instance, the target reaching is activated to lead the movement towards the destination. However, the UGV has been obstructed by an obstacle who is randomly located at the coordinates of (1.1, 1.12). Hence, the obstacle avoidance algorithm is activated to avoid collision. It is noticeable that the turning has occurred in the right direction to avoid clashing with the obstacle. The turning implies changing the orientation of the UGV. Consequently, the UGV requires to switch back to the target reaching algorithm. The new orientation of the UGV is determined by using the magnetic compass module. The angle of the target is calculated based on a trigonometric approach between the coordinates of the actual position of the UGV and the target point. The angle difference of the target reaching and the UGV's heading governs the new required orientation towards the target. Fig. 8.12(b) illustrates the turning process around the obstructing obstacle. After a successful avoidance of the hindering obstacle by controlling the speed of the wheels, the UGV makes a feasible movement towards the target. Finally, Fig. 8.12(c) demonstrates the competing of the

navigation process by reaching the coordinates of the target point. The travelled distance in this case takes longer to reach the target comparing to case study I. As a result, the elapsed time is increased to 2.6 seconds.




(a)

(b)

(c)

Fig. 8.12 Case study-II when one obstacle is existed; (a) at the starting point, (b) at the turning point and (c) at the target point.

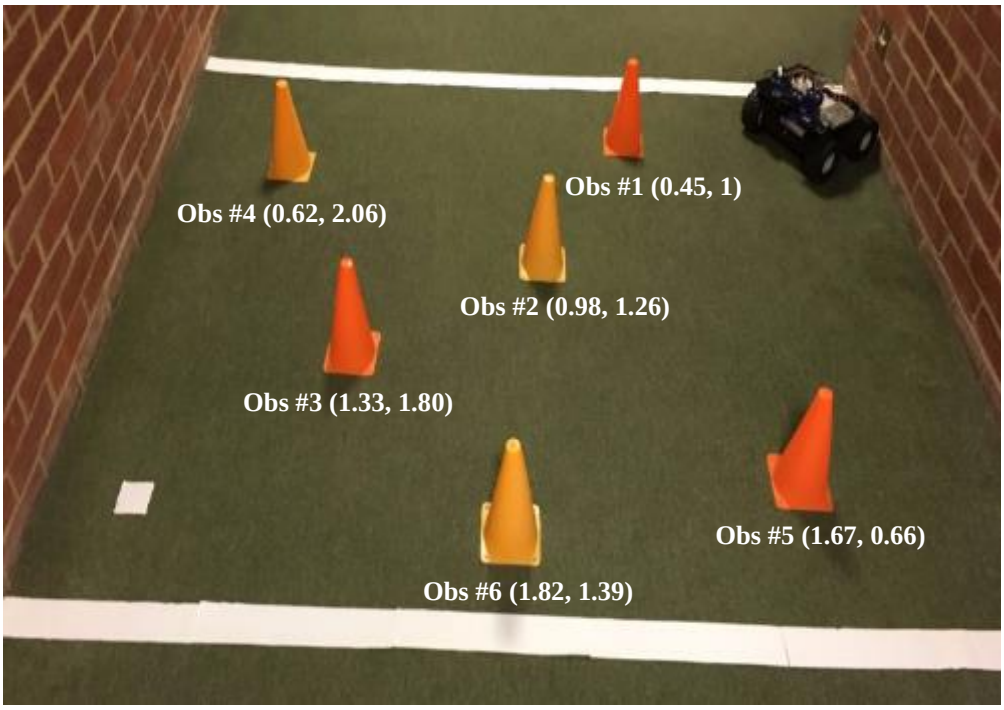
### *8.8.3 Case Study-III*

In this case study, multiple constructing obstacles are placed randomly within the workspace. Although the number of obstructing obstacles is six, only two obstructing obstacles are practically hindering the movement of the UGV. In theory, the principle of operation in terms of obstacle avoidance and target reaching will be invariant when the number of obstacles is increased. Instead, the switching frequency between the obstacle avoidance and target reaching algorithms will be increased. When the UGV avoids an obstructing obstacle, it intends to change back its direction towards the target point. Therefore, if another obstructing obstacle confront the movement of the UGV, this in turn applies a challenge for the UGV in order to adapt its localisation and move forwards the goal. Reasonable distances are given to separate the obstacles to enable the UGV of making the best decision. If the obstacles are completely blocking the movement path, the stopping decision will occur to avoid collision, the UGV will resume its movement when the path is clear of obstructing obstacles.

Similarly, Fig. 8.13(a) demonstrates the case study III when the workspace is filled by six obstructing obstacles. It also shows the measured coordinates of each obstructing obstacle. The UGV starts the movement from the same aforementioned starting point. Fig. 8.13(b) shows the

first turning that occurs due to obstructing obstacle no.2. The UGV turns left to avoid this obstacle, afterward, it changes the heading towards the target. However, obstacle no.3 has lightly hindered the movement of the UGV. Hence, it has slightly changed its heading again accordingly as illustrated in Fig. 8.13(c). Then, it has also changed back its orientation in the direction of the goal point. Such process has validated a robust switching frequency between the target reaching and obstacle avoidance algorithms. Fig. 8.13(d) demonstrates the last instance of the target reaching after an effective and a successful avoidance of obstructing obstacles. The elapsed time equals 2.8 seconds for completing the navigation process in this case study.

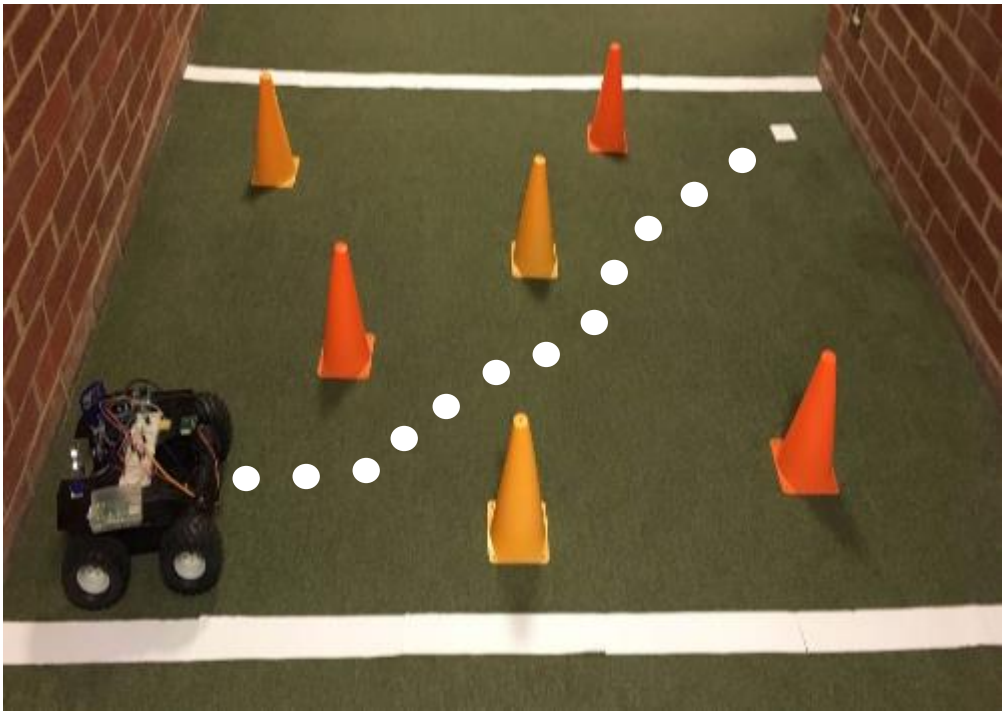
A simple comparison of the three case studies demonstrates that the travelled distance and the elapsed time are taken longer in case study III. Obviously, the case study I shows the shortest distance and elapsed time. The case study II is slightly different by adding an obstructing obstacle in the path. Thus, longer travelled distance and elapsed time are occurred in comparing to the case study I.




(a)

(b)

(c)



(d)

Fig. 8.13 Case study-III when one obstacles are existed; (a) at the starting point, (b) at the first turning point, (c) at the second turning point and (d) at the target point.

## 8.9 Chapter Summary

An unmanned architecture based on a real-time implementation is introduced for an unmanned ground vehicle. The system uses multiple sensors to sense its surrounding and localise itself simultaneously with a workspace. The distances between obstructing obstacles and the UGV's platform are measured by three ultrasonic range sensors. The orientation of the UGV has been controlled using a magnetic compass module. The travelled distances are determined based on quadratic encoders. For each sensing device, an algorithm is developed to create communication links between the sensing devices and the control unit of the UGV on one hand, and between the sensing devices and an environment of the other hand. The integrated platform based on the embedded system is investigated using three main case studies to demonstrate the interaction between the UGV and the environment. The real time experiments have demonstrated that the proposed algorithms have accomplished the requirements of the design effectively and feasibly with fast processing time.

# Chapter 9

# Conclusions and Directions for Future Work

## 9.1 Conclusions

A robust control technique is proposed to address the problem of trajectory tracking of an unmanned ground vehicle. This technique utilizes fractional-order proportional integral derivative controllers to control the motion of the UGV to track the behaviour of predefined reference trajectories. The trajectories are divided into two different categories i.e. continuous and non-continuous gradient trajectories. The heading control of the UGV has been accomplished based on controlling the movement of the left and right wheels using the proposed FOPID controllers. The implemented model of the non-holonomic unmanned ground vehicle takes into consideration both kinematic and dynamic and actuation characteristics. Each FOPID controller composes of five parameters i.e. proportional, integral, derivative and the fractional order of both integral and derivative. The values of the parameters are vital for obtaining an optimal response of a system. Therefore, a PSO algorithm is used to tune and optimize the parameters of the FOPID controller. The criterion of obtained the optimal parameters is based on minimizing the cost function used in the PSO algorithm. The effectiveness of the proposed FOPID controllers has been verified through different trajectories as given in Chapter 4 using MATLAB–Simulink software package.

In addition, the stability of the fractional-order system has been investigated to prove that the proposed controllers are stable at the operating conditions. Moreover, the robustness of the entire system is examined by applying different sources of disturbances. The obtained results of FOPID controller demonstrate the advantage of the proposed FOPID controllers in terms of minimising trajectory tracking error and the completing of the path following. The performance of the constructed model has shown a robustness for changes due to the applied disturbances even it applies permanently. In order to validate the preference of the proposed FOPID, many comparisons have been conducted to compare the proposed FOPID model with a conventional PID controller. The simulation results have feasibly demonstrated that the proposed FOPID has improved the operation performance by minimising the trajectory tracking error and reducing control efforts.

Although the fractional order PID controller for a trajectory tracking of an UGV has been successfully designed, the design has not been significantly capable of minimising the tracking error in a non-continuous gradient trajectory in particular. Hence, another controller is proposed based on artificial neural networks. This controller is implemented based on Fractional Order Proportional Integral Derivative controller designed earlier to track a non-continuous gradient trajectory. The driven inputs of the UGV are the right and left motor voltage. The outputs represent the steering orientation and the actual velocity of the UGV. Therefore, two artificial neural network controllers are designed to control the inputs of the UGV. In order to train the two neural controllers, Levenberg-Marquardt algorithm is used for obtaining the parameters of the NN.

The newly developed NN has been compared with the FOPID controller to demonstrate the effectiveness of the introduced approach. The obtained results of the artificial intelligent neural technique reveal a significant improvement in term of minimizing trajectory-tracking error and improving control actions over the proposed FOPID controller. Accordingly, the combination of NN and FOPID has been improved a better efficiency and performance with respect to the standalone FOPID and the traditional PID controllers. Moreover, the combination NN-FOPID has been demonstrated as fast learning capability to track the given trajectories. It is observable that the smoothness and faster convergence performance of the tracking error for vehicle velocity and orientation angle have been satisfied.

The motion control of the UGV is essential in the industry of automation. Particularly in dynamic environment, it is required to have a control methodology that is efficiently capable to drive the UGV without collision with surrounding objects. In this thesis, we have proposed fuzzy inference systems to achieve the navigation in different cluttered dynamic environments. The structure of the fuzzy inference systems is based on two controllers. The first controller uses three sensors based on the distances from the front, the right and the left. The second controller employs the angle difference between the heading of the vehicle and the targeted angle to choose an optimal route based on dynamic environments and reach a desired destination with minimum running power and time. The first controller is used for the target reaching and is called abbreviated to TR-FIS controller and the second controller is utilised for the obstacle avoidance and is abbreviated to OA-FIS controller. The TR-FIS controller is proposed to ensure that the UGV reaches its target destination by keeping its heading direction towards the targeted destination when there is no obstacle approaching the UGV. The input of this controller is the angle difference (AD).

The OA-FIS controller is responsible for changing the angular velocity of the driving wheels to provide an obstacle manoeuvring when any obstacle approaches the UGV's platform. The inputs of this controller are the sensory information that have been gathered from the UGV's sensors. The ultrasonic sensors are attached to the UGV to calculate e the distance of any obstacle that approaches the UGV from the front, right and left. The control and navigation architectures have been demonstrated in four different scenarios. In the first scenario, six moving obstacles are used; they all have similar sizes and velocities. In the second scenario, the six obstacles are assumed to have similar sizes, but they move at dissimilar velocities. In the third scenario, the obstacles are constructed to have different sizes but they all have similar velocities. In the four scenario, one static and five dynamic obstacles are placed in the same workspace to demonstrate the adaptation and performance of the UGV in a more realistic scenario.

In all scenarios, the designed target reaching and obstacle avoidance controllers have proved their capability of avoiding obstacles and guiding the UGV towards its final destination successfully. Therefore, the fuzzy inference systems proved to be a satisfactory control methodology for UGV to avoid static and moving obstacles in busy and dynamic environments. It has offered an intelligent behaviour in facing the uncertainty issues that are presented by such dynamic environments. The simulation results have been carried out in four different scenarios to investigate the validation and effectiveness of the introduced controllers of the fuzzy inference system. The reported simulation results are conducted using MATLAB software package. The results demonstrate that the fuzzy inference systems consistently perform the manoeuvring task and route planning efficiently even in complex environments with populated static and dynamic obstacles. Our methodology has been compared to a state of the art, where similar approach was used. The comparison results have approved that the developed FIS has been successfully improved the navigation performance of the UGV in terms of generating an optimal path and minimising the elapsed time.

Furthermore, we have made efforts to design a new adaptive neuro-fuzzy inference system for navigation and obstacle avoidance. It has been inspired based on the FIS to demonstrate a new approach that combines the learning capability and fast convergence for an obstacle avoidance and a target reaching. The adaptive neuro-fuzzy inference system consists of four standalone ANFIS controllers, two of which are used for regulating both the left and right angular velocities of the UGV in order to reach the target position, and other two ANFIS controllers are used for optimal heading adjustment in order to avoid obstacles. The two velocity controllers receive three sensor inputs: front distance, right distance and left distance

for the low-level motion control. Two heading controllers deploy the angle difference between the heading of UGV and the angle to the target to choose the optimal direction.

The simulation experiments have been carried out into two case studies to investigate the feasibility of the proposed ANFIS technique. In the first case study, seven identical static obstacles are placed randomly within the workspace. In the second case, eleven obstacles are positioned, but different sizes and shapes are used. In case of reducing obstacle numbers significantly, the obstacle avoidance ANFIS controller does not activate until the vehicle confronts an obstacle. Therefore, the target reaching ANFIS controller is activated most of the time. This case will be similar to the situation when a path is free of obstacles. Thus, the UGV will traverse through the shortest straight line that connects between both start and target points. In contrast, in multiple obstacles simulation, the obstacle avoidance ANFIS controller is activated more frequently in order to avoid obstacles. After completing each avoidance, the UGV switches to the target reaching ANFIS controller. In both considered case studies, the UGV is masterful of avoiding obstacles safely and reaching the target with a feasible and smooth online-generated path between the initial and the target points. The simulation results have been presented using MATLAB software package, showing that ANFIS can perform the navigation and path-planning task safely and efficiently in a workspace populated with unknown obstacles. In addition, our ANFIS model based navigation is compared with a recent ANFIS model conducted in the state of the art. It has been clearly found that our ANFIS model has performed much better feasible and optimal path. Therefore, the elapsed time has been reduced notably.

The architecture of the UGV is practically assembled validate the proposed algorithms based on real time experiments. The assembly involves three main categories of sensor technology. First, three ultrasonic sensors are used in three directions i.e. left, front and right to sense surrounding obstacles approaching the UGV's platform. In addition, the magnetic compass module sensor is utilised to determine the orientation of the UGV whilst navigation towards its destination. Moreover, a quadratic encoder sensor is applied to the shaft of motors to measure the travelled distance, hence, the localisation of the UGV can be calculated accordingly. The experimental results are conducted based on three different case studies. They all have demonstrated an effective response in terms of obstacle avoidance and target reaching.

## 9.2 Directions for Future Work

Overall, in this thesis, the research objectives are successfully met. The thesis has extended the theory of solved the trajectory tracking and navigation by formalising and solving reconstruction problems and presenting new proposed techniques for industrial automation. The formalisation of the reconstruction case applications and workplace scenarios have been validated through the development of the proposed techniques. A number of novel results as well as problems associated with the developed formalisation have been discovered. Consequently, it has been noticed that some results can be improved further based on either proposed new algorithms or developing the existing algorithms. In addition, some limitations are identified in our work. The non-continuous gradient trajectories still need further research to design a controller methodology that reduces the tracking error to zero. It is also observable that a sharp response occurs at turnings whilst navigation. Nevertheless, a smoother switching mechanism is still needed to provide a feasible and smooth response for driving the movement of the UGV. This reveals that it is possible to extend the work for investigation new constructed problems and developing algorithms that are more efficient. Hence, several possible directions for future research can be proposed. They include extending formalisation of event reconstruction, developing more efficient event reconstruction algorithm, investigating new ways of constructing system models, and developing practical applications of the results of this work. Whilst this thesis has demonstrated the motion control of the UGV safely and efficiently. Notwithstanding, many opportunities and much research for extending the scope and topics can be considered and addressed for future directions as follows:

1- For trajectory tracking, signifcant improvements have been made to reduce the tracking error and enhance the performance of the system based on the proposed control methodologies. Nonetheless, it is still a noticeable tracking error in a variety of trajectories. Therefore, new control architectures can be proposed to diminish the tracking error to the zero level.

2- Some parameters of the proposed controllers are optimised based particle swarm optimisation and Levengberg-Marquardt algorithms. Thus, different optimising algorithms can be proposed such as ant colony optimisation and simulated annealing algorithms. They might provide better results.

3- The fitness functions of the PSO and LM algorithms are implemented based on the integral square error and mean square error criteria, respectively. However, the fitness functions can be constructed based on different criteria such as the integral of time

multiplied by absolute error criterion and integral of the absolute magnitude of the error criterion.

4- For navigation and obstacle avoidance, construct three-dimensional environments are important case study to be investigated. This can be fulfilled by considering 3D structure and the height of objects and UGV in a navigation task. Less height objects can represent the motion of children in highly dynamic environments. Hence, they should have been given special arrangements and priority to be avoided.
5- The switching mechanism demonstrates a sharp and a rapid response at some occasions. Hence, it requires a developed switching mechanism that provides a smoother response.
6- To avoid the complexity and the delay in a computational time of fuzzy inference systems, it has been used a small number of inference rules. As a result, the velocity of the wheels shows acute responses at turning points. Therefore, more probabilities can be studied and then the system can be optimised to obtain an utmost response and performance.
7- The navigation is conducted using FIS and ANFIS controllers. Therefore, different techniques and optimisation algorithms can be combined and proposed to optimise the followed path and the elapsed time further.
8- Multiple waypoints can be set up between the initial and the target points. This makes the UGV navigate through multiple destinations at certain times.
9- The trajectory tracking and navigation have been addressed individually in this work. Therefore, both problems can be integrated together to track a particular trajectory and avoid obstacles instantaneously if any is existed.
10- Vision sensors can be embedded in the designed architecture to aid navigation processes.
11- Design a multi-agent architecture for navigation of several unmanned ground vehicles to navigate and operate in complex scenarios.

# References


Abdessemed, F., Faisal, M., Emmadeddine, M., Hedjar, R., Al-Mutib, K., Alsulaiman, M. and Mathkour, H. (2014), "A Hierarchical Fuzzy Control Design for Indoor Mobile Robot", *International Journal of Advanced Robotic Systems*, Vol. 11 No. 1, pp. 1–16.

Aboelela, M. a. S., Ahmed, M.F. and Dorrah, H.T. (2012), "Design of aerospace control systems using fractional PID controller", *Journal of Advanced Research*, Cairo University, Vol. 3 No. 3, pp. 225–232.

Achour, N. and Chaalal, M. (2011), "Mobile Robots Path Planning using Genetic Algorithms", *The Seventh International Conference on Autonomic and Autonomous Systems*, Venice, Italy, pp. 111–115.

Adeli, H., Tabrizi, M.H.N., Mazloomian, A., Hajipour, E. and Jahed, M. (2011), "Path Planning for Mobile Robots using Iterative Artificial Potential Field Method", *International Journal of Computer Science Issues*, Vol. 8 No. 4, pp. 28–32.

Aenugu, V. and Woo, P.-Y. (2012), "Mobile Robot Path Planning with Randomly Moving Obstacles and Goal", *International Journal of Intelligent Systems and Applications*, Vol. 4 No. 2, pp. 1–15.

Aldair, A.A. and Wang, W.J. (2010), "Design of Fractional Order Controller Based on Evolutionary Algorithm for a Full Vehicle Nonlinear Active Suspension Systems", *International Journal of Control and Automation*, Vol. 3 No. 4, pp. 33–46.

Algabri, M., Mathkour, H. and Ramdane, H. (2014), "Mobile Robot Navigation and Obstacle-avoidance using ANFIS in Unknown Environment", *International Journal of Computer Applications*, Vol. 91 No. 14, pp. 36–41.

Antonelli, G., Member, S., Chiaverini, S. and Fusco, G. (2007), "A Fuzzy-Logic-Based Approach for Mobile Robot Path Tracking", *IEEE Transactions on Fuzzy Systems*, Vol. 15 No. 2, pp. 211–221.

Arbo, M.H., Raijmakers, P.A. and Mladenov, V.M. (2014), "Applications of neural networks for control of a double inverted pendulum", *12th Symposium on Neural Network Applications in Electrical Engineering*, IEEE, Belgrade, Serbia, pp. 89–92.

Barraquand, J. and Latombe, J.-C. (1989), "On nonholonomic mobile robots and optimal maneuvering", *The IEEE International Symposium on Intelligent Control*, pp. 340–347.

Berg, M. de, Kreveld, M. Van, Overmars, M. and Schwarzkopf, O. (2000), *Computaional Geometry: Algorithms and Applications*, 2nd ed., Springer-Verlag Berlin Heidelberg, Germany.

Bhatti, S.A., Malik, S.A. and Daraz, A. (2016), "Comparison of P-I and I-P controller by using Ziegler-Nichols tuning method for speed control of DC motor", *International Conference on Intelligent Systems Engineering*, IEEE, Islamabad, Pakistan, pp. 330–334.

Borenstein, J., Everett, H.R., Feng, L. and Wehe, D. (1997), "Mobile robot positioning: Sensors and techniques", *Journal of Robotic Systems*, Vol. 14 No. 4, pp. 231–249.

Branson, J.S. and Lilly, J.H. (2001), "Incorporation, characterization, and conversion of negative rules into fuzzy inference systems", *IEEE Transactions on Fuzzy Systems*, Vol. 9 No. 2, pp. 253–268.

Cao, J. and Cao, B. (2006), "Design of Fractional Order Controllers Based on Particle Swarm Optimization", *1st IEEE Conference on Industrial Electronics and Applications*, IEEE, Singapore, pp. 1 – 6.

Cao, J., Liang, J.I.N. and Cao, B. (2005), "Optimization of Fractional Order PID Controllers Based on Genetic Algorithms", *The Fourth International Conference on Machine Learning and Cybernetics*, IEEE, Guangzhou, pp. 5686–5689.

Cardenas, A.M., Razuri, J.G., Sundgren, D. and Rahmani, R. (2013), "Autonomous Motion of Mobile Robot Using Fuzzy-Neural Networks", *12th Mexican International Conference on Artificial Intelligence*, IEEE, Mexico, pp. 80–84.

Chaudhary, G. and Ohri, J. (2016), "3-DOF Parallel Manipulator Control using PID Controller", *1st IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems*, pp. 1–6.

Chen, Y., Petr, I. and Xue, D. (2009), "Fractional Order Control - A Tutorial", *American Control Conference*, Hyatt Regency Riverfront, MO, USA, pp. 1397–1411.

Cherni, F., Boutereaa, Y., Rekik, C. and Derbel, N. (2016), "Path Planning for mobile robots using fuzzy logic controller in the presence of static and moving obstacles", *The 3rd International Conference on Automation, Control, Engineering and Computer Science*, pp. 503–509.

Chi, K.-H. and Lee, M.-F.R. (2011), "Obstacle Avoidance in Mobile Robot using Neural Network", *The IEEE International Conference on Consumer Electronics, Communications and Networks*, IEEE, Xianning, China, pp. 5082–5085.

Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. and Thrun, S. (2005), *Principles of Robot Motion: Theory, Algorithms, and Implementations*, A Bradford Book, Cambridge, Massachusetts.

Cong, Y.Z. and Ponnambalam, S.G. (2009), "Mobile robot path planning using ant colony optimization", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, IEEE, Singapore, pp. 851–856.

Cui, S., Su, X., Zhao, L., Bing, Z. and Yang, G. (2010), "Study on Ultrasonic Obstacle Avoidance of Mobile Robot Based on Fuzzy Controller", *The IEEE International Conference on Computer Application and System Modeling*, IEEE, Tianjin, China, pp. 233–237.

Dadios, E.P. (2012), *Fuzzy Logic – Controls, Concepts, Theories and Applications*, InTech, Croatia.

Davies, T. and Jnifene, A. (2006), "Multiple Waypoint Path Planning for a Mobile Robot using Genetic Algorithms", *IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems*, pp. 12–14.

Deshpande, S.U. and Bhosale, S.S. (2013), "Adaptive neuro-fuzzy inference system based robotic navigation", *IEEE International Conference on Computational Intelligence and Computing Research*, IEEE, Enathi, pp. 1–4.

Dong, T., Liao, X.H., Zhang, R., Sun, Z. and Song, Y.D. (2005), "Path Tracking and Obstacle Avoidance of UAVs - Fuzzy Logic Approach", *The 14th IEEE International Conference on Fuzzy Systems*, IEEE, Reno, NV, pp. 43–48.

Doroftei, I., Grosue, V. and V.S. (2007), *Omnidirectional Mobile Robot – Design and Implementation: Bioinspiration and Robotics Walking and Climbing Robots*, *I-Tech Education and Publishing*, available at:https://doi.org/10.5772/60142.

Driankov, D. and Saffiotti, A. (2001), *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag Heidelberg, New York.

Du, X., Chen, H. and Gu, W. (2005), "Neural network and genetic algorithm based global path planning in a static environment", *Journal of Zhejiang University- SCIENCE A*, Vol. 6 No. 6, pp. 549–554.

Elshamli, A., Abdullah, H. a. and Areibi, S. (2004), "Genetic algorithm for dynamic path planning", *Canadian Conference on Electrical and Computer Engineering*, IEEE, Niagara Falls, Ontario, pp. 677–680.

Engedy, I. and Horváth, G. (2010), "Artificial Neural Network based Local Motion Planning of a Wheeled Mobile Robot", *11th International Symposium on Computational Intelligence and Informatics*, IEEE, Budapest, pp. 213–218.

Faisal, M., Hedjar, R., Al, M. and Al-Mutib, K. (2013), "Fuzzy Logic Navigation and Obstacle Avoidance by a Mobile Robot in an Unknown Dynamic Environment", *International Journal of Advanced Robotic Systems*, Vol. 10 No. 37, pp. 1–7.

Fausett, L. (1993), *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*, 1st ed., Pearson.

Fierro, R. and Lewis, F.L. (1997), "Control of a Nonholonomic Mobile Robot : Backstepping Kinematics into Dynamics", *Journal of Robotic Systems*, Vol. 14 No. 3, pp. 149–163.

Fierro, R. and Lewis, F.L. (1998), "Control of a nonholonomic mobile robot using neural networks.", *IEEE Transactions on Neural Networks*, Vol. 9 No. 4, pp. 589–600.

Fukao, T., Nakagawa, H. and Adachi, N. (2000), "Adaptive tracking control of a nonholonomic mobile robot", *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, Vol. 16 No. 5, pp. 609–615.

Ganapathy, V., Yun, S.C., Member, S. and Ng, J. (2009), "Fuzzy and Neural Controllers for Acute Obstacle Avoidance in Mobile Robot Navigation", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Singapore, pp. 1236–1241.

Garrido, S., Moreno, L., Blanco, D. and Jurewicz, P. (2011), "Path Planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching", *International Journal of Robotics and Automation*, Vol. 2 No. 1, pp. 42–64.

Ge, S.S. and Cui, Y.J. (2002), "Dynamic Motion Planning for Mobile Robots Using Potential Field Method", *Autonomous Robots*, Vol. 13 No. 3, pp. 207–222.

Ghorbani, A., Shiry, S. and Nodehi, A. (2009), "Using Genetic Algorithm for a Mobile Robot Path Planning", *International Conference on Future Computer and Communication*, IEEE, Kuala Lumpar, Malaysia, pp. 164–166.

Guo, L., Ge, P.-S., Yue, M. and Zhao, Y.-B. (2014), "Lane Changing Trajectory Planning and Tracking Controller Design for Intelligent Vehicle Running on Curved Road", *Mathematical Problems in Engineering*, pp. 1–9.

Hagan, M.T. and Menhaj, M.B. (1994), "Training feedforward networks with the Marquardt algorithm", *IEEE Transactions on Neural Networks*, Vol. 5 No. 6, pp. 2–6.

Hajar, S., Mohammad, A., Jeffril, M.A. and Sariff, N. (2013), "Mobile Robot Obstacle Avoidance By Using Fuzzy Logic Technique", *The 3rd IEEE International Conference on System Engineering and Technology*, IEEE, Shah Alam, Malaysia, pp. 331–335.

Hamani, M. and Hassam, A. (2012), "Mobile Robot Navigation in Unknown Environment Using Improved APF Method", *The 13th Intrenational Arab Conference on Information Technology*, Algeria, pp. 453–458.

Hao, Y.U., Gong-you, T., Hao, S.U., Chun-peng, T. and Jian, Z. (2014), "Trajectory Tracking Control of Wheeled Mobile Robots via Fuzzy Approach", *33rd Chinese Control Conference (CCC)*, IEEE, Nanjing, pp. 8444–8449.

He, K., Sun, H. and Cheng, W. (2008), "Application of fuzzy neural network based on T-S model for mobile robot to avoid obstacles", *7th World Congress on Intelligent Control and Automation*, IEEE, Chongqing, China, pp. 8282–8285.

Holmberg, R. and Khatib, O. (1999), "Development and Control of a Holonomic Mobile Robot for Mobile Manipulation Tasks", *International Journal of Robotics Research*, Vol. 19 No. 11, pp. 1066–1074.

Islam, M.S., Azim, M.A., Jahan, M.S. and Othman, M. (2006), "Design and Synthesis of Mobile Robot Controller using Fuzzy", *The IEEE International Conference on Semiconductor Electronics*, IEEE, Kuala Lumpur, Malaysia, pp. 825–829.

Jang, J.R. (1993), "ANFIS : Adaptive Network Based Fuzzy Inference System", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23 No. 3, pp. 665–685.

Janglová, D. (2004), "Neural Networks in Mobile Robot Motion", *Inernational Journal of Advanced Robotic Systems*, Vol. 1 No. 1, pp. 15–22.

Jiang, Z.-P. and Nijmeijer, H. (1997), "Tracking Control of Mobile Robots: A Case Study in Backstepping", *Automatica*, Vol. 33 No. 97, pp. 1393–1399.

Joshi, M.M. and Zaveri, M. (2010), "Neuro-Fuzzy Based Autonomous Mobile Robot Navigation System", *11th Int. Conf. Control, Automation, Robotics and Vision*, IEEE, Singapore, pp. 384–389.

Jung, I.-K., Hong, K.-B., Hong, S.-K. and Hong, S.-C. (1999), "Path planning of mobile robot using neural network", *Proceedings of the IEEE International Symposium on Industrial Electronics*, IEEE, Slovenia, pp. 979 – 983.

Keighobadi, J., Menhaj, M.B. and Kabganian, M. (2010), "Feedback-linearization and fuzzy controllers for trajectory tracking of wheeled mobile robots", *Kybernetes*, Vol. 39 No. 1, pp. 83–106.

Khalid, M. and Omatu, S. (1992), "A neural network controller for a temperature control system", *IEEE Control Systems*, Vol. 12 No. 3, pp. 58–64.

Khelchandra, T., Huang, J. and Debnath, S. (2014), "Path Planning of Mobile Robot with Neuro-Genetic-Fuzzy Technique in Static Environment", *International Journal of Hybrid Intelligent Systems*, Vol. 11 No. 2, pp. 71–80.

Koren, Y. and Borenstein, J. (1991), "Potential field methods and their inherent limitations for mobile robot navigation", *International Conference on Robotics and Automation*, Sacramento. California, pp. 1398–1404.

Lee, C.C.C. (1990), "Fuzzy logic in control systems: fuzzy logic controller, Part II", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20 No. 2, pp. 404–418.

Lee, C.-H. and Chang, F.-K. (2010), "Fractional-order PID controller optimization via improved electromagnetism-like algorithm", *Expert Systems with Applications*, Elsevier Ltd, Vol. 37 No. 12, pp. 8871–8878.

Lee, J.-W., Choi, B.-S., Park, K.-T. and Lee, J.-J. (2011), "Comparison between heterogeneous ant colony optimization algorithm and Genetic Algorithm for global path planning of mobile robot", *The International Symposium on Industrial Electronics*, IEEE, Gdansk, Poland, pp. 881–886.

Li, X. and Choi, B. (2013), "Design of Obstacle Avoidance System for Mobile Robot using Fuzzy Logic Systems", *International Journal of Smart Home*, Vol. 7 No. 3, pp. 321–328.

Liang, Y., Xu, L., Wei, R. and Hu, H. (2010), "Adaptive Fuzzy Control for Trajectory Tracking of Mobile Robot", *The IEEE International Conference on Intelligent Robots and Systems*, IEEE, Taipei, Taiwan, pp. 4755–4760.

Mamdani, E.H. (1977), "Application of fuzzy logic to approximate reasoning using linguistic synthesis", *IEEE Transactions on Computers*, Vol. C-26 No. 12, pp. 1182–1191.

Mamdani, E.H. and Assilian, S. (1975), "An experiment in linguistic synthesis with a fuzzy logic controller", *International Journal of Man-Machine Studies*, Vol. 7 No. 1, pp. 1–13.

Miao, H. and Tian, Y. (2008), "Robot Path Planning in Dynamic Environments Using a Simulated Annealing Based Approach", *The 10th International Conference on Control, Automation, Robotics and Vision*, IEEE, Hanoi, Vietnam, pp. 17–20.

Mohamed, M.J. (2013), "Obstacles Avoidance for Mobile Robot Using Enhanced Artificial Potential Field", *Al-Khwarizmi Engineering Journal*, Vol. 9 No. 1, pp. 71–82.

Mohanta, J.C., Parhi, D.R. and Patel, S.K. (2011), "Path planning strategy for autonomous mobile robot navigation using Petri-GA optimisation", *Computers & Electrical Engineering*, Elsevier Ltd, Vol. 37 No. 6, pp. 1058–1070.

Mohareri, O. (2009), *Mobile Robot Trajectory Tracking Using Neural Networks*, American University of Sharjah.

Monje, C.A., Chen, Y., Vinagre, B.M., Xue, D. and Feliu, V. (2010), *Fractional-Order Systems and Controls Fundamentals and Applications*, edited by Grimble, M.J. and Johnson, M.A., Springer London Dordrecht Heidelberg, New York.

Nedelkovski, D. (2015), "Ultrasonic Sensor HC-SR04 and Arduino Tutorial", *Howtomechatronics*, available at: http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/ (accessed 26 June 2017).

Nguyen, D.H. and Widrow, B. (1990), "Neural Networks for Self-Learning Control Systems", *IEEE Control Systems*.

Nie, X. (2011), "Application of neural network for thermal error compensation in CNC machine tool", *The 2nd International Conference on Computing, Control and Industrial Engineering*, IEEE, Wuhan, China, pp. 211–215.

Oscar. (2013), "Connect Raspberry Pi and Arduino with Serial USB Cable", *Oscarliang*, available at: https://oscarliang.com/connect-raspberry-pi-and-arduino-usb-cable/ (accessed 27 June 2017).

Padhy, P.K., Sasaki, T., Nakamura, S. and Hashimoto, H. (2010), "Modeling and position control of mobile robot", *The 11th IEEE International Workshop on Advanced Motion Control*, IEEE, Nagaoka, Japan, pp. 100–105.

Parhi, D.R. (2008), *Neuro-Fuzzy Navigation Technique for Control of Mobile Robots, Motion Planning*, edited by Jing, X.-J., InTech.

Pawlowski, S., Dutkiewicz, P., Kozlowski, K. and Wroblewski, W. (2001), "Fuzzy logic implementation in mobile robot control", *Proceedings of the 2nd International Workshop on Robot Motion and Control.*, IEEE, Bukowy Dworek, pp. 65–70.

Pedro, J.O., Dangor, M. and Kala, P.J. (2016), "Differential Evolution-Based PID Control of a Quadrotor System for Hovering Application", *The IEEE Congress on Evolutionary Computation*, pp. 2791–2798.

Podlubny, I. (1999), "Fractional-order systems and PI/sup /spl lambda//D/sup /spl mu//-controllers", *IEEE Transactions on Automatic Control*, Vol. 44 No. 1, pp. 208–214.

Pradhan, S.K., Parhi, D.R. and Panda, A.K. (2009), "Fuzzy logic techniques for navigation of several mobile robots", *Applied Soft Computing*, Vol. 9 No. 1, pp. 290–304.

Raguraman, S.M., Tamilselvi, D. and Shivakumar, N. (2009), "Mobile Robot Navigation Using Fuzzy Logic Controller", *The IEEE International Conference on Control, Automation, Communication and Energy Conservation*, IEEE, Perundurai, Tamilnadu, pp. 1–5.

Raja, P. and Pugazhenthi, S. (2009), "Path Planning for Mobile Robots in Dynamic Environments Using Particle Swarm Optimization", *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, IEEE, Kottayam, Kerala, pp. 401–405.

Ramezanian, H. and Balochian, S. (2013), "Optimal Design a Fractional-Order PID Controller using Particle Swarm Optimization Algorithm", *International Journal of Control and Automation*, Vol. 6 No. 4, pp. 55–68.

Rashid, R., Elamvazuthi, I., Begam, M. and Arrofiq, M. (2010), "Differential Drive Wheeled Mobile Robot (WMR) Control Using Fuzzy Logic Techniques", *The Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation*, IEEE, Kota Kinabalu, Malaysia, pp. 51–55.

Rosli, N., Ibrahim, R., Ismail, I., Hassan, S.M. and Chung, T.D. (2016), "Neural Network Architecture Selection for Efficient Prediction Model of Gas Metering System", *The 2nd IEEE International Symposium on Robotics and Manufacturing Automation*, IEEE, Ipoh, Perak, Malaysia.

Sadati, N., Zamani, M. and Mahdavian, H.R.F. (2006), "Hybrid particle swarm-based-simulated annealing optimization techniques", *The 32nd Annual Conference on Industrial Electronics*, Paris, France, pp. 644–648.

Sedighi, K.H., Ashenayi, K., Manikas, T.W. and Wainwright, R.L. (2004), "Autonomous Local Path Planning for a Mobile Robot Using a Genetic Algorithm", *IEEE Proceedings of the Congress on Evolutionary Computation*, IEEE, Potland, USA, pp. 1338–1345.

Shi, E., Cai, T. and He, C. (2007), "Study of the New Method for Improving Artifical Potential Field in Mobile Robot Obstacle Avoidance", *Proceedings of the IEEE International Conference on Automation and Logistics*, Jinan, China, pp. 282–286.

Shi, P. and Cui, Y. (2010), "Dynamic path planning for mobile robot based on genetic algorithm in unknown environment", *Chinese Control and Decision Conference*, IEEE, Xuzhou, China, pp. 4325–4329.

Shojaei, K., Tarakameh, A. and Shahri, A.M. (2009), "Adaptive Trajectory Tracking of WMRs based on Feedback Linearization Technique", *The International Conference onMechatronics and Automation*, IEEE, Changchun, China, pp. 729–734.

Siegward, R., Nourbakhsh, I.R. and Scaramuzza, D. (2011), *Introdction to Autonomous Mobile Robot*, edited by Arkin, R., 2nd ed., The MIT Press Cambridge, Massachusetts, Cambridge, Massachusetts.

Simon, D. (2013), *Evolutionary Optimization Algorithms*, 1 edition., Wiley & Sons, Inc., Hoboken, New Jersey.

Solea, R., Filipescu, A. and Nunes, U. (2009), "Sliding-Mode Control for Trajectory-Tracking of a Wheeled Mobile Robot in Presence of Uncertainties", *IEEE Asian Control Conference (ACC)*, IEEE, Hong Kong, pp. 1701 – 1706.

Sumathi, S. and Paneerselvam, S. (2010), *Computational Intelligence Paradigms: Theory and Applications Using MATLAB*, Taylor & Francis Group.

Tang, L., Dian, S., Gu, G., Zhou, K., Wang, S. and Feng, X. (2010), "A Novel Potential Field Method for Obstacle Avoidance and Path Planning of Mobile Robot", *The 3rd IEEE International Conference on Computer Science and Information Technology*, IEEE, Chengdu, China, pp. 633–637.

Tewolde, G.S. (2013), "Evolutionary Learning for Improving Performance of Robot Navigation", *IEEE International Conference on Electro-Information Technology*, IEEE, Lincoln, NE, USA, pp. 1–5.

Tian, Y., Wang, Q., Wang, Y. and Jin, Q. (2014), "A novel design method of multi-objective robust PID controller for industrial process", *The 2014 9th IEEE Conference on Industrial Electronics and Applications*, pp. 242–246.

Tuncer, A. and Yildirim, M. (2012), "Dynamic path planning of mobile robots with improved genetic algorithm", *Computers & Electrical Engineering*, Elsevier Ltd, Vol. 38 No. 6, pp. 1564–1572.

Turevskiy, A. (2016), "Control System Toolbox Overview", *MathWorks*.

Ugalde-loo, C.E., Liceaga-castro, E. and Liceaga-castro, J. (2005), "2x2 Individual Channel Design MATLAB® Toolbox", *44th IEEE Conference on Decision and Control*, IEEE, Seville, Spain, pp. 7603–7608.

Uszkoreit, M.P.H., Wahlster, M.V.W., Wooldridge, M.J., Buchanan, B.G., Hayes, P.J., Hendler, J.A., Jennings, N., et al. (2007), *Robot Cognition and Navigation*, Springer Berlin Heidelberg New York.

Velagic, J., Osmic, N. and Lacevic, B. (2010), "Design of Neural Network Mobile Robot Motion Controller", in Ramov, B. (Ed.), *New Trends in Technologies*, InTech, p. 242.

Vivero, O. and Liceaga-Castro, J. (2008), "MIMO Toolbox for Matlab", *Student Paper, 2008 Annual IEEE Conference*, Aalborg, pp. 1 – 5.

Vukosavljev, S.A., Kukolj, D., Papp, I. and Markoski, B. (2011), "Mobile robot control using combined neural- fuzzy and neural network", *12th IEEE International Symposium on Computational Intelligence and Informatics*, IEEE, Budapest, Hungary, pp. 351–356.

Wilamowski, B.M. (2009), "Neural Network Architectures and Learning Algorithms", *IEEE IIndustrial Electronics*, Vol. 3 No. 4, pp. 56–63.

Wilamowski, B.M. and Irwin, J. david (Eds.). (2011), *The Industrial Electronics Handbook: Intelligent Systems*, 2nd ed., Taylor and Francis Group, LLC.

Xie, M., Li, L. and Wang, Z. (2012), "Trajectory tracking control for mobile robot based on the fuzzy sliding mode", *The 10th World Congress on Intelligent Control and Automation*, IEEE, Beijing, China, pp. 2706–2709.

Xu, Q., Kan, J., Chen, S. and Yan, S. (2014), "Fuzzy PID Based Trajectory Tracking Control of Mobile Robot and its Simulation in Simulink", *International Journal of Control and Automation*, Vol. 7 No. 8, pp. 233–244.

Ye, J. (2008), "Adaptive control of nonlinear PID-based analog neural networks for a nonholonomic mobile robot", *Neurocomputing*, Vol. 71 No. 7-9, pp. 1561–1565.

Ye, J. (2013), "Tracking control of two-wheel driven mobile robot using compound sine function neural networks", *Connection Science*, Vol. 25 No. 2-3, pp. 139–150.

Yongjie, Z., Jiang, C. and Shuguo, W. (2002), "A new path-planning algorithm for mobile robot based on neural network", *Conference on Computers, Communications, Control and Power Engineering*, IEEE, Beijing, China, pp. 1570–1573.

Yun, S.C., Ganapathy, V. and Chong, L.O. (2010), "Improved Genetic Algorithms based Optimum Path Planning for Mobile Robot", *The 11th Int. Conf. Control, Automation, Robotics and Vision*, IEEE, Singapore, pp. 1565–1570.

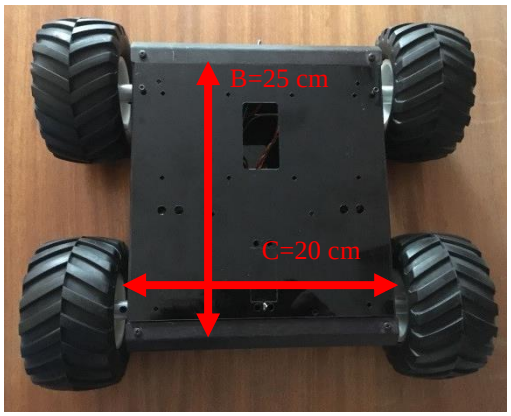
Zamani, M., Karimi-Ghartemani, M., Sadati, N. and Parniani, M. (2009), "Design of a fractional order PID controller for an AVR using particle swarm optimization", *Control Engineering Practice*, Elsevier, Vol. 17 No. 12, pp. 1380–1387.

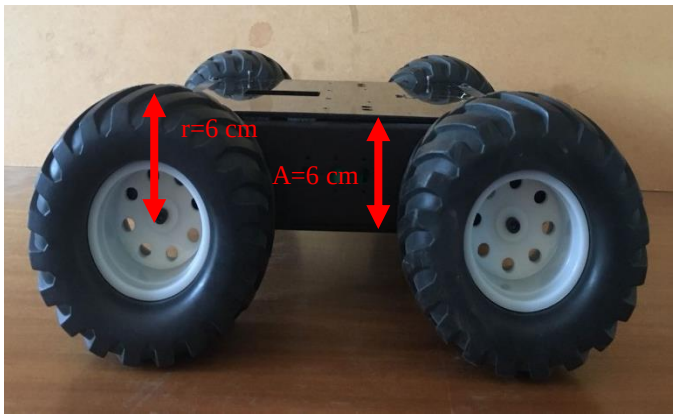
Zhang, Y., Gong, D. and Zhang, J. (2013), "Robot path planning in uncertain environment using multi-objective particle swarm optimization", *Neurocomputing*, Elsevier, Vol. 103, pp. 172–185.

Zhao, T. and Wang, Y. (2012), "A Neural-Network Based Autonomous Navigation System Using Mobile Robots", *The 12th International Conference on Control Automation Robotics & Vision*, IEEE, Guangzhou, China, pp. 1101–1106.

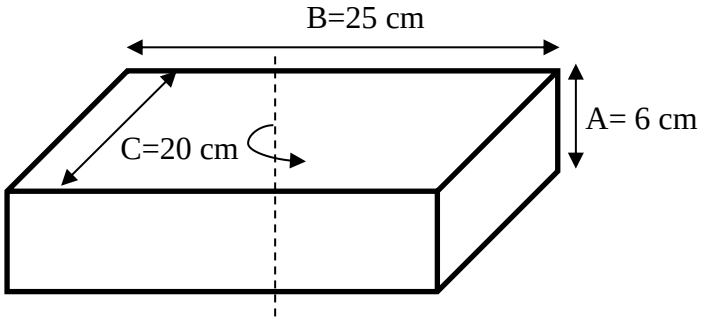
Zhao, Y. and Gu, J. (2013), "Robot Path Planning Based on Improved Genetic Algorithm", *Proceeding of the IEEE International Conference on Robotics and Biomimetics*, IEEE, Shenzhen, China, pp. 2515–2522.

# Appendix: Calculation of moment of inertia

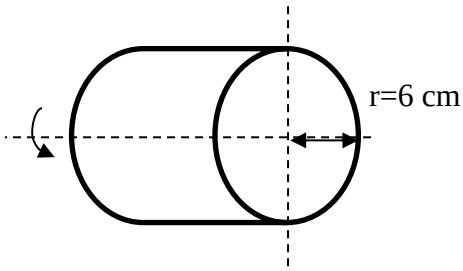



The axis of rotation passes through the centre of gravity.




$$I_c = \frac{m}{12}(B^2 + C^2)$$




$$I_c = \frac{mr^2}{2}$$

| **Parameter** | | m | $A, B, C$ | r | $I_c$ |
|---|---|---|---|---|---|
| **Description** | | Mass | Dimensions | Radius | Moment of inertia |
| **Vehicle's body** | **Value** | 5 | 6,25,20 | - | 0.0427 |
| **Wheel** | **Value** | 0.20 | - | 6 | 0.00036 |
| **Unit** | | kg | cm | cm | $kgm^2$ |